反辞書の
効率的な構築手法に
関する研究

Study on Efficient Algorithms for Construction of an Antidictionary.

深江 裕忠

電気通信大学大学院
情報システム学研究科

博士（工学）の学位申請論文

2014年3月

反辞書の効率的な構築手法に関する研究

Study on Efficient Algorithms for Construction of an Antidictionary.

博士論文審査委員会

審査主査：　森田 啓義　教授
審査委員：　長岡 浩司　教授
審査委員：　笠井 裕之　准教授
審査委員：　古賀 久志　准教授
審査委員：　多田 好克　教授

Abstract

An antidictionary is a data structure to represent minimal forbidden words which never appear on a given string but its proper substrings do appear on it.

Since Crochemore et al. originally introduced an antidictionary for a lossless data compression in 2000, it has been widely applied as a useful tool to many applications, data compression of ECG, frame-synchronization, branch prediction, and so on.

Although a naïve method for constructing an antidictionary takes time and memory complexity which is a quadratic proportional to length of a string, some algorithms with linear time and memory complexity for construction of an antidictionary have been proposed so far.

However, in spite of their liner complexity, those conventional algorithms consume a huge amount of computational time and space resources to construct an antidictionary in practical applications.

In this thesis, we discuss efficient algorithms to construct an antidictionary.

First we propose an off-line algorithm by using two array structures, that is, suffix array and L-array. A suffix array is a sorted list of the suffixes of a given string while an L-array is a list of length of the longest common prefix between adjacent pairs of suffixes on the suffix array.

Computer simulation shows that time and space complexities of the proposed algorithm are about 1/20 and 1/2.5 lower than a conventional linear construction algorithm, respectively.

Next, we consider a problem how to construct dynamically an antidictionary automaton which accepts minimum forbidden words of a given string.

Each time a new symbol of the string is entered into the system, we update the antidictionary automaton to fit the intermediate input data. By identifying which minimum forbidden word is deleted from or added to the current automaton, we obtain an on-line algorithm to construct an antidictionary automaton of a given string.

概要

情報技術の発展により，近年では様々な機器にセンサーが搭載され，センサーで検知したデータに従って動作する機器が多くなってきた．センサーは，温度，振動，超音波，電波など様々なものがあるが，取得するデータは数値の系列である．この数値の系列（入力系列）に対して，検出や解析を行うために，入力系列に出現する全ての部分系列を登録したデータベース (ここでは辞書と呼ぶ) がよく使われる．

本論文で研究対象にしている反辞書とは，従来の一般的な辞書とは異なり，入力系列に出現しない系列（極小禁止語）を登録する辞書である．極小禁止語とは，それ自身は入力系列に出現しないが，極小禁止語の真の部分系列は全て入力系列に出現する系列である．すなわち，入力系列に出現しない系列の中で極小性をもつ．反辞書は，全ての部分系列を登録する辞書よりも，登録する系列が少ないという性質をもつ．また，反辞書を構成するデータ構造として，極小禁止語を受理するオートマトンや，接尾辞木を拡張して極小禁止語のノードを追加するなど効率的な検索方法が提案されている．

反辞書は，Crochemore らが 2000 年に提案した無歪みデータ圧縮法に初めて用いられ，符号化で使用される確率モデルの作成において，その有効性が示された．データ圧縮以外にも，反辞書は，分岐予測，同期符号，不整脈検出などへ応用されている．

しかし，反辞書を構築するには，計算量が入力系列長に比例することが知られているが，実用上は，膨大な記憶量と計算時間を必要とする．例えば，太田・森田によって提案された接尾辞木を用いた反辞書構築手法では，アルファベットサイズが多値の場合，記憶量が膨大になる．計算機実験では，アルファベットサイズが 256 の場合，入力系列長の約 1 万倍の記憶量を必要とした．

そこで，本論文では，記憶量と計算時間を改善する新しい反辞書構築手法を二つ提案する．

一つ目は，入力系列を全て入力後に反辞書を構築する静的な手法である．二つ目は，入力系列を逐次的に読み込みながら，反辞書を変化させていく動的な手法である．構築手法を考える場合，一般的には，動的な手法よりも静

的な手法の方が簡単である．そこで，最初に静的な手法について検討を行い，その成果を動的な手法へ活用する．

静的な手法では，新しい反辞書の別表現を提案する．この別表現では，極小禁止語を，先頭記号と後続する系列に分けて考えることにより，あと1記号先頭に加えることで極小禁止語になる系列が，入力系列の接尾辞配列ならびに高さ配列（本論文ではL配列と呼ぶ）によって特徴づけられること，すなわち，先頭に１記号加えれば極小禁止語になる系列は，接尾辞配列上の接尾辞の先頭部分に存在し，その長さはL配列で求めることができることを示す．ここでL配列とは，接尾辞配列上で隣接した二つの接尾辞に共通する最長接頭辞の長さの配列である．

さらに，本論文では，被覆集合を提案する．被覆集合とは，入力系列の任意の部分系列が，接尾辞配列上の接尾辞の先頭部分に存在している範囲を表現する集合である．接尾辞配列上の接尾辞は辞書式順序昇順で整列しているので，被覆集合の範囲を比較することで，入力系列の部分系列同士の辞書式順序昇順の大小関係を判別できる．すなわち，この2つの系列の先頭から順に記号を比較する必要がなく，定数時間で系列の比較が行えるようになる．

この反辞書の別表現と被覆集合を用いることで，接尾辞配列，L配列をそれぞれ3回走査するだけで，先頭に１記号加えれば極小禁止語になる系列をすべて含む高々2n+2個（nは入力系列長）の系列を辞書式順序昇順で求められる算法を提案する．提案方法においては，極小禁止語は，利用した被覆集合の範囲をそのまま活用して，被覆集合の範囲と系列長，そして，先頭に加える記号を要素に持つ配列で表現される．本論文では，この極小禁止語の配列表現を反辞書配列と呼んでいる．

そして，計算機実験によって提案した算法の有効性を確かめたところ，接尾辞木を用いた従来手法と比較して，計算時間が約20分の1，記憶量が約2.5分の1に改善されることが分かった．

動的な手法では，反辞書そのものではなく，反辞書オートマトン，すなわち，反辞書に含まれる極小禁止語を受理するオートマトンを，入力系列を逐次的に読み込みながら動的に構築する手法を提案する．

まず，すでに読み込まれた入力系列の末尾に新たな記号が加わることにより，反辞書がどのように更新されるかを明らかにした．申請者が修士論文で得た知見である，反辞書の更新において，ある特定の極小禁止語が一つ削除されることと，新たに加わる極小禁止語（新規極小禁止語）は，その両端のどちらか一方に削除される極小禁止語を含む，という二つの性質に加え，本論文では，新規極小禁止語の長さを削除される極小禁止語の長さで上と下か

ら評価した.

つぎに, 反辞書の別表現と同様に, あと1記号加えることで新規極小禁止語になる系列を, 入力系列の末尾から探索する. さらに, 更新前の反辞書オートマトンを用いて, 探索した系列を新規極小禁止語にするために加える記号を求める.

これらの結果を用いることによって, 反辞書オートマトンの構築するために, 従来の手法では, 接尾辞を表す木構造やオートマトンを経由して作成する必要があったのが, その手間を省いて, 入力系列から直接構築する手法を提案する.

目 次

第1章　はじめに **12**
1.1　研究背景 12
1.2　研究のねらい 13

第2章　定義とデータ構造 **15**
2.1　有限アルファベット集合と系列 15
2.2　辞書 15
2.3　接尾辞と接頭辞 16
2.4　極小禁止語と反辞書 17
2.5　辞書式順序比較 17
2.6　接尾辞配列 18
2.7　共通接頭辞の最大長 19
2.8　L配列 19
2.9　被覆集合 20
2.10 接頭記号と接尾記号 21

第3章　接尾辞木を用いた反辞書構築 **23**
3.1　反辞書構築のアプローチ 23
3.2　接尾辞木 24
3.3　suffix link と MF–link 26
3.4　反辞書の構築 27
3.4.1　タイプIの極小禁止語の追加 27
3.4.2　タイプIIの極小禁止語の追加 28

第4章　配列を用いた静的反辞書構築の提案 **30**
4.1　接尾辞木を用いた反辞書構築手法の記憶量の内訳 30
4.2　反辞書の別表現 30
4.3　極小禁止語の先頭文字を削除した系列 31
4.4　提案手法の概要 34

4.5 系列の表現 35
4.6 Step 3 のアルゴリズム 37
4.7 Step 4 のアルゴリズム 40
4.8 計算時間の評価 44
4.9 記憶量の評価 45
4.10 計算機実験 47

第 5 章 動的反辞書構築の提案 51
5.1 静的反辞書構築と動的反辞書構築 51
5.2 表記の拡張 51
5.3 反辞書オートマトン 52
5.4 既知の結果 54
5.5 新しく加わる極小禁止語の長さ 56
5.6 反辞書の変化と提案アルゴリズム 58

第 6 章 おわりに 61

図 目 次

2.1 $\boldsymbol{x}$ = MISSISSIPPI のときの接尾辞配列 18
2.2 $\boldsymbol{x}$ = MISSISSIPPI のときの L 配列 19
2.3 $\boldsymbol{x}$ = MISSISSIPPI のときの S の被覆集合 20
2.4 $\boldsymbol{x}$ = MISSISSIPPI のときの S の接尾辞配列と接頭記号 22

3.1 これまでの反辞書の構築方法 23
3.2 $\boldsymbol{x}$ = aababacbcb のときの接尾辞トライ (上) と接尾辞木 (下) 25
3.3 接尾辞の suffix link（実線矢印）と MF–link（点線矢印） 26
3.4 接尾辞木へのタイプ I の反辞書の追加例 28
3.5 接尾辞木へのタイプ II の反辞書の追加例 29

4.1 計算時間の比較（実線：提案手法，点線：接尾辞木を用いた手法） . . 48
4.2 記憶量の比較（実線：提案手法，点線：接尾辞木を用いた手法） . . . 48
4.3 アルファベットサイズ m ごとの計算時間 50
4.4 アルファベットサイズ m ごとの記憶量 50

5.1 $\boldsymbol{x}^{10}$ = abcaababac の反辞書オートマトン．黒丸のノードが極小禁止語を示す． 53

表 目 次

4.1　$\boldsymbol{x} = \mathtt{ababb}$ のときの $\bar{\boldsymbol{t}}_i$ と $\bar{\boldsymbol{b}}_i$（$\bar{\boldsymbol{b}}_i$ の—は，存在しない事を示す）．　．　33

記法一覧

$\mathcal{X}$	有限アルファベット集合．	p.15
$\boldsymbol{x}_i^j$	系列 $x_i x_{i+1} \cdots x_j$．	p.15
λ	空系列．	p.15
$\boldsymbol{x}$	長さ n の系列 $x_1 x_2 \cdots x_n$（第 4 章までの記法）．	p.15
$\boldsymbol{x}^n$	長さ n の系列 $x_1 x_2 \cdots x_n$（第 5 章以降の記法）．	p.52
$\mathcal{D}(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の辞書．	p.15
$\mathcal{S}(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の全ての接尾辞の集合．	p.16
$\sigma(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の先頭記号を削る接尾辞演算子．	p.16
$\sigma^k(\boldsymbol{x}^n)$	接尾辞演算子を系列 $\boldsymbol{x}^n$ に k 回繰り返しほどこす．	p.16
$\mathcal{P}(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の全ての接頭辞の集合．	p.16
$\pi(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の末尾記号を削る接頭辞演算子．	p.16
$\pi^k(\boldsymbol{x}^n)$	接頭辞演算子を系列 $\boldsymbol{x}^n$ に k 回繰り返しほどこす．	p.16
$\mathcal{A}(\boldsymbol{x}^n)$	系列 $\boldsymbol{x}^n$ の反辞書．	p.17
$\boldsymbol{v} \prec \boldsymbol{w}$	辞書式順序昇順での大小関係．$\boldsymbol{v}$ の方が $\boldsymbol{w}$ より小さい．	p.17
$\boldsymbol{r}_i$	辞書式順序昇順で i 番目の接尾辞．なお，$\boldsymbol{r}_0 = \lambda$ である．	p.18
S	系列 $\boldsymbol{x}^n$ の接尾辞配列．	p.18
$\mathsf{S}[i]$	系列 $\boldsymbol{x}^n$ の接尾辞配列の i $(0 \leq i \leq n)$ 番目の要素．	p.18
$\perp$	接尾辞配列の境界を示す仮想記号．	p.18
$\mathrm{lcp}\,(\boldsymbol{v}, \boldsymbol{w})$	系列 $\boldsymbol{v}$ と $\boldsymbol{w}$ の共通接頭辞の最大の系列長．	p.19

L	系列 $\boldsymbol{x}^n$ のL配列．……p.19
$\mathsf{L}[i]$	系列 $\boldsymbol{x}^n$ のL配列の i $(-1 \leq i \leq n)$ 番目の要素．……p.19
$\mathcal{C}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ の被覆集合．……p.20
$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ の接頭記号の全ての集合（第4章までの記法）．……p.21
$\mathcal{X}_n^{\mathrm{H}}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ の接頭記号の全ての集合（第5章以降の記法）．……p.52
$\mathcal{X}^{\mathrm{T}}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ の接尾記号の全ての集合（第4章までの記法）．……p.21
$\mathcal{X}_n^{\mathrm{T}}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ の接尾記号の全ての集合（第5章以降の記法）．……p.52
$\mathcal{N}(\boldsymbol{v})$	系列 $\boldsymbol{v}$ のローカス……p.26
$\boldsymbol{L}(p)$	ノード p のラベル列．……p.26
$\mathcal{S}(p)$	ノード p のsuffix link先ノード．……p.26
$\mathcal{H}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ に対する，$\pi(\boldsymbol{u})$ と $\boldsymbol{u}$ の全接頭記号集合の差集合（第4章までの記法）．……p.30
$\mathcal{H}_n(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ に対する，$\pi(\boldsymbol{u})$ と $\boldsymbol{u}$ の全接頭記号集合の差集合……p.52
$\mathcal{T}(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ に対する，$\sigma(\boldsymbol{u})$ と $\boldsymbol{u}$ の全接尾記号集合の差集合（第4章までの記法）．……p.31
$\mathcal{T}_n(\boldsymbol{u})$	系列 $\boldsymbol{x}^n$ の部分系列 $\boldsymbol{u}$ に対する，$\sigma(\boldsymbol{u})$ と $\boldsymbol{u}$ の全接尾記号集合の差集合……p.52
$\mathcal{G}_{\mathrm{H}}$	系列 $\boldsymbol{x}^n$ の全ての極小禁止語の先頭文字を削除した系列の集合．p.31
$\bar{\boldsymbol{t}}_i$	系列 $\boldsymbol{x}^n$ の極小禁止語の先頭文字を削除した系列の候補．長さを $\mathsf{L}[i-1]$ で求めている．……p.33

$\bar{\boldsymbol{b}}_i$ 系列 $\boldsymbol{x}^n$ の極小禁止語の先頭文字を削除した系列の候補．長さを $\mathsf{L}[i]$ で求めている．. p.33

$\mathcal{G}$ 全ての $\bar{\boldsymbol{t}}_i$ と $\bar{\boldsymbol{b}}_i$ の集合．$\mathcal{G}_{\mathrm{H}}$ を含んでいる．. p.34

$\mathcal{G}_{\mathrm{T}}$ 全ての $\bar{\boldsymbol{t}}_i$ の集合．. p.37

$\mathcal{G}_{\mathrm{B}}$ 全ての $\bar{\boldsymbol{b}}_i$ の集合．. p.37

$\boldsymbol{t}_i$ 集合 $\mathcal{G}_{\mathrm{T}}$ の要素を辞書式順序昇順に整列したときの i $(0 \leq i \leq n)$ 番目の系列．. p.37

$\boldsymbol{b}_i$ 集合 $\mathcal{G}_{\mathrm{B}}$ の要素を辞書式順序昇順に整列したときの i $(0 \leq i \leq |\mathcal{G}_{\mathrm{B}}| - 1)$ 番目の系列．. p.37

$\boldsymbol{g}_i$ 集合 $\mathcal{G}$ の要素を辞書式順序昇順に整列したときの i $(0 \leq i \leq |\mathcal{G}| - 1)$ 番目の系列．. p.37

第1章　はじめに

1.1　研究背景

反辞書とは，従来の一般的な辞書とは異なり，入力系列に出現しない系列（極小禁止語）を登録する辞書である．Crochemoreらが2000年に提案した反辞書法[5]で，歪みのないデータ圧縮に有効であることが明らかにされた．データ圧縮以外にも，分岐予測[6]，同期符号[7]，不整脈検出[8]などの応用が提案されている．

反辞書を利用するには，反辞書から構築した反辞書オートマトンを用いる．例えば，反辞書法の場合，符号側では，入力系列にしたがってオートマトンの状態を遷移していき，遷移先が一意に定まるときは記号を削除することでデータを圧縮する．そして，復号側では，符号側から反辞書と生成した圧縮データが与えられる．与えられた反辞書から反辞書オートマトンを生成し，圧縮データにしたがって反辞書オートマトンを遷移することでデータを復元する．このとき，符号側で削除した記号は遷移先が一意に決まるので，データは完全に復元される．

すなわち，反辞書法は符号化側と復号側の処理がオートマトンを遷移するだけなので，計算機への負荷が少ないという利点がある．この利点は，組込み機器などの資源が限られ，記憶量と通信路容量を十分確保できない環境下において，優れた圧縮手法となる可能性を秘めている．また，反辞書法以外についても，反辞書オートマトンを用いるので計算機への負荷が少ない．

反辞書の構築は，太田・森田によって，入力系列長に比例する，すなわち，線形の計算時間と記憶量で構築する手法[9]が提案されている．この手法では，入力記号系列の接尾辞を木構造で表現した接尾辞木を用いる．接尾辞木へMF–linkと呼ばれる特別なリンクを追加して反辞書を構築する．ただし，アルファベットが多値の場合，記憶量が膨大になるという問題がある．これは，MF–linkが使用する記憶量は線形だが，アルファベットサイズにも比例するために係数が大きくなるのが原因である．

そこで，本論文ではより少ない計算時間と記憶量で反辞書を構築すること

を検討する．

1.2 研究のねらい

本論文では，記憶量と計算時間を改善する新しい反辞書構築手法を2つ提案する．1つ目は，入力系列を全て入力後に反辞書を構築する静的な手法である．2つ目は，入力系列を逐次的に読み込みながら，反辞書を変化させていく動的な手法である．構築手法を考える場合，一般的には，動的な手法よりも静的な手法のほうが簡単である．そこで，最初に静的な手法について検討を行い，その成果を動的な手法へ活用する．

1つ目の静的手法では，接尾辞を配列で表現する接尾辞配列 [10] に注目する．ただし，単純に接尾辞木の代わりに接尾辞配列を用いると，記憶量は削減できるが，入力系列長 n に対して，n^2 に比例する計算時間を必要とする．また，接尾辞配列に高さ配列 [11],[12]（本論文ではL配列と呼ぶ）を利用した方法でも平均して $n \log n$ に比例する計算時間を必要とする問題がある．ここでL配列とは，接尾辞配列上で隣接した二つの接尾辞に共通する最長接頭辞の長さの配列である．

そこで，本論文では計算時間を線形とするために，あと1記号を先頭に加えると反辞書の要素になる系列を接尾辞配列から探索し，反辞書を構築する手法を提案する．

そのために，極小禁止語を先頭記号と後続する系列に分けて考える，反辞書の新しい別表現を提案する．そして，あと1記号先頭に加えることで極小禁止語になる系列が，入力系列の接尾辞配列ならびにL配列によって特徴づけられること，すなわち，先頭に1記号加えれば極小禁止語になる系列は，接尾辞配列上の接尾辞の先頭部分に存在し，その長さはL配列で求めることができることを示す．

さらに，本論文では，被覆集合を提案する．被覆集合とは，入力系列の任意の部分系列が，接尾辞配列上の接尾辞の先頭部分に存在している範囲を表現する集合である．接尾辞配列上の接尾辞は辞書式順序昇順で整列しているので，被覆集合の範囲を比較することで，入力系列の部分系列同士の辞書式順序昇順の大小関係を判別できる．すなわち，この2つの系列の先頭から順に記号を比較する必要がなく，定数時間で系列の比較が行えるようになる．

この反辞書の別表現と被覆集合を用いることで，接尾辞配列，L配列をそれぞれ3回走査するだけで，先頭に1記号加えれば極小禁止語になる系列をすべて含む高々$2n+2$個の系列を辞書式順序昇順で求められる算法を提案す

る．提案方法においては，極小禁止語は，利用した被覆集合の範囲をそのまま活用して，被覆集合の範囲と系列長，そして，先頭に加える記号を要素に持つ配列で表現される．本論文では，この極小禁止語の配列表現を反辞書配列と呼んでいる．

また，計算機実験を行い，接尾辞木を用いた従来手法と比較して，計算時間が約 20 分の 1，記憶量が約 2.5 分の 1 に改善されることを示す．

2 つ目の動的な手法では，反辞書そのものではなく，反辞書オートマトン，すなわち，反辞書に含まれる極小禁止語を受理するオートマトンを，入力系列を逐次的に読み込みながら動的に構築する手法を提案する．

そのために，入力系列の末尾に新たな記号が加わることで反辞書がどのように更新するのかを検討する．

これまでの研究で，反辞書が変化するときに，削除される極小禁止語と新たに加える極小禁止語（新規極小禁止語）は明らかになっている．しかし，効率的に求めることができない問題があった．本論文では，さらに検討を進め，新たに加わる極小禁止語の長さを評価する．

つぎに，反辞書の別表現と同様に，あと 1 記号加えることで新規極小禁止語になる系列を，入力系列の末尾から探索する．さらに，更新前の反辞書オートマトンを用いて，探索した系列を新規極小禁止語にするために加える記号を求める．

これらの結果を用いることによって，反辞書オートマトンの構築するために，従来の手法では，接尾辞を表す木構造やオートマトンを経由して作成する必要があったのが，その手間を省いて，入力系列から直接構築する手法を提案する．

第2章　定義とデータ構造

2.1　有限アルファベット集合と系列

有限アルファベット集合を

$$\mathcal{X} = \{\xi_1, \xi_2, \ldots, \xi_m\}\ (\xi_1 < \xi_2 < \cdots < \xi_m, m \geq 2)$$

とおき，$\mathcal{X}$ 上の系列を，

$$\boldsymbol{x}_i^j = x_i x_{i+1} \cdots x_j\ (x_k \in \mathcal{X}, 1 \leq i \leq k \leq j)$$

とする．

ここで，集合のサイズと系列長は $|\cdot|$ で表記する．例えば，$|\mathcal{X}| = m, |\boldsymbol{x}_i^j| = j - i + 1$ である．なお，系列長 0 を空系列とし，λ と表記する．便宜上，$i > j$ のとき $\boldsymbol{x}_i^j = \lambda$ とする．また，系列長 $n(\geq 1)$ の系列 $\boldsymbol{x}_1^n$ を $\boldsymbol{x}$ と表記する．

2.2　辞書

辞書 (dictionary) $\mathcal{D}(\boldsymbol{x})$ とは，$\boldsymbol{x}$ 上に出現する部分系列の全ての集合であり，

$$\mathcal{D}(\boldsymbol{x}) = \{\boldsymbol{x}_i^j | 1 \leq i \leq j \leq n\} \cup \{\lambda\}$$

とする．なお，$\mathcal{D}(\lambda) = \{\lambda\}$ であることを注記する．

例 1 系列 $\boldsymbol{x} = \mathsf{abbca}$ の辞書は，

$$\mathcal{D}(\boldsymbol{x}) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{ab}, \mathsf{bb}, \mathsf{bc}, \mathsf{ca}, \mathsf{abb}, \mathsf{bbc}, \mathsf{bca}, \mathsf{abbc}, \mathsf{bbca}, \mathsf{abbca}, \lambda\}$$

となる．

2.3 接尾辞と接頭辞

系列 $\boldsymbol{x} = \boldsymbol{x}^n$ に対して $\boldsymbol{x}$ の接尾辞 (suffix) とは，$\boldsymbol{x}_i^n$ $(1 \leq i \leq n)$ に空系列 λ を加えたものである．ここで，$\boldsymbol{x}$ の全ての接尾辞からなる集合 $\mathcal{S}(\boldsymbol{x})$ は，

$$\mathcal{S}(\boldsymbol{x}) = \{\boldsymbol{x}_i^n | 1 \leq i \leq n\} \cup \{\lambda\}$$

と表される．さらに，接尾辞演算子 σ を $\sigma(\boldsymbol{x}) = \boldsymbol{x}_2^n$ と定義する．$\boldsymbol{x}$ に σ を k (≥ 1) 回繰り返し施すと，

$$\sigma^k(\boldsymbol{x}) = \boldsymbol{x}_{1+k}^n$$

となる．すなわち，$\sigma^k(\boldsymbol{x})$ は，$\boldsymbol{x}$ の先頭から k 記号分削除した系列である．

また，$\boldsymbol{x}_1^i$ $(1 \leq i \leq n)$ と空系列 λ を $\boldsymbol{x}$ の接頭辞 (prefix) と呼ぶ．ここでは，$\boldsymbol{x}$ の全ての接頭辞からなる集合 $\mathcal{P}(\boldsymbol{x})$ は，

$$\mathcal{P}(\boldsymbol{x}) = \{\boldsymbol{x}_1^i | 1 \leq i \leq n\} \cup \{\lambda\}$$

と表される．さらに，接頭辞演算子 π を $\pi(\boldsymbol{x}) = \boldsymbol{x}_1^{n-1}$ と定義する．$\boldsymbol{x}$ に π を k (≥ 1) 回繰り返し施すと，

$$\pi^k(\boldsymbol{x}) = \boldsymbol{x}_1^{n-k}$$

となる．すなわち，$\pi^k(\boldsymbol{x})$ は，$\boldsymbol{x}$ の末尾から k 記号分削除した系列である．

なお，$\sigma(\lambda) = \lambda, \pi(\lambda) = \lambda$ であることを注記する．

例 2 系列 $\boldsymbol{x} = \mathtt{abbca}$ の全ての接尾辞の集合 $\mathcal{S}(\boldsymbol{x})$ と，全ての接頭辞の集合 $\mathcal{P}(\boldsymbol{x})$ は，

$$\begin{aligned} \mathcal{S}(\boldsymbol{x}) &= \{\mathtt{abbca}, \mathtt{bbca}, \mathtt{bca}, \mathtt{ca}, \mathtt{a}, \lambda\}, \\ \mathcal{P}(\boldsymbol{x}) &= \{\mathtt{abbca}, \mathtt{abbc}, \mathtt{abb}, \mathtt{ab}, \mathtt{a}, \lambda\} \end{aligned}$$

となる．

また，

$$\begin{aligned} &\sigma(\boldsymbol{x}) = \mathtt{bbca}, \quad \sigma^2(\boldsymbol{x}) = \mathtt{bca}, \\ &\pi(\boldsymbol{x}) = \mathtt{abbc}, \quad \pi^2(\boldsymbol{x}) = \mathtt{abb} \end{aligned}$$

となる．

2.4 極小禁止語と反辞書

系列 $\boldsymbol{x}$ 上に出現しない系列を禁止語とよぶ．そして，系列 $\boldsymbol{v}$ が次の (2.1), (2.2), (2.3) のすべてを満すとき，$\boldsymbol{v}$ は $\boldsymbol{x}$ の極小禁止語 (Minimal Forbidden Word) である．

$$\boldsymbol{v} \notin \mathcal{D}(\boldsymbol{x}), \tag{2.1}$$

$$\pi(\boldsymbol{v}) \in \mathcal{D}(\boldsymbol{x}), \tag{2.2}$$

$$\sigma(\boldsymbol{v}) \in \mathcal{D}(\boldsymbol{x}). \tag{2.3}$$

したがって，極小禁止語は $\boldsymbol{x}$ 上に出現しないが，極小禁止語の端を削除した系列は $\boldsymbol{x}$ 上に出現している．

反辞書 (antidictionary) $\mathcal{A}(\boldsymbol{x})$ とは，系列 $\boldsymbol{x}$ に対する極小禁止語のすべての集合である．すなわち，

$$\mathcal{A}(\boldsymbol{x}) = \{\boldsymbol{v} \notin \mathcal{D}(\boldsymbol{x}) \,|\pi(\boldsymbol{v}) \in \mathcal{D}(\boldsymbol{x}), \sigma(\boldsymbol{v}) \in \mathcal{D}(\boldsymbol{x})\}$$

となる．

例 3 有限アルファベット集合 $\mathcal{X} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$，系列 $\boldsymbol{x} = \mathtt{abbca}$ の反辞書は，

$$\mathcal{A}(\boldsymbol{x}) = \{\mathtt{aa}, \mathtt{ac}, \mathtt{ba}, \mathtt{cb}, \mathtt{cc}, \mathtt{abc}, \mathtt{bbb}, \mathtt{cab}\}$$

となる．

2.5 辞書式順序比較

系列 $\boldsymbol{v}, \boldsymbol{w}$ の辞書式順序昇順での大小関係を，比較記号 $\prec$ で示すことにする．$\boldsymbol{v} \prec \boldsymbol{w}$ は，

$$(v_1 < w_1) \vee (v_1 = w_1) \wedge (\sigma(\boldsymbol{v}) \prec \sigma(\boldsymbol{w}))$$

と定義する．便宜上，$\lambda < \xi_1$ とする．

例 4 次の 5 つの系列 $\mathtt{bb}, \mathtt{aca}, \mathtt{c}, \mathtt{abba}, \mathtt{abb}$ の辞書式順序昇順での大小関係は，

$$\mathtt{abb} \prec \mathtt{abba} \prec \mathtt{aca} \prec \mathtt{bb} \prec \mathtt{c}$$

となる．

2.6 接尾辞配列

系列$\boldsymbol{x}$の全ての接尾辞を辞書式順序昇順で並べたとき，$i\,(0 \leq i \leq n)$番目の接尾辞をランクiの接尾辞と呼び，$\boldsymbol{r}_i$と表記する．すなわち，$\boldsymbol{r}_0 = \lambda$となり，$i > 0$のときは，

$$\boldsymbol{r}_i = \min\{\boldsymbol{v} \in \mathcal{S}(\boldsymbol{x}) \,|\boldsymbol{r}_{i-1} \prec \boldsymbol{v}\} \tag{2.4}$$

となる．

系列$\boldsymbol{x}$の接尾辞配列 (suffix array) Sとは，Sのi番目の要素が

$$\mathsf{S}[i] = n - |\boldsymbol{r}_i| + 1$$

で与えられる配列である．この$\mathsf{S}[i]$は接尾辞$\boldsymbol{r}_i$の$\boldsymbol{x}$上におけるインデックスを表す．よって，$\boldsymbol{x}$の左端から$\mathsf{S}[i]-1$個の記号を削ると，$\boldsymbol{r}_i$になるので，

$$\boldsymbol{r}_i = \sigma^j(\boldsymbol{x})\ (j = \mathsf{S}[i] - 1)$$

が成り立つ．

なお，便宜上，本論では境界を示すために，$\mathcal{X}$に属さない記号$\perp$を用いて$\boldsymbol{r}_{-1} = \boldsymbol{r}_{n+1} = \perp$とおく．

例 5 図2.1に，$\boldsymbol{x} = \mathtt{MISSISSIPPI}$（系列長11）の接尾辞配列の概要を示す．

最初に，$\boldsymbol{x}$の全ての接尾辞を辞書式順昇順で整列する．整列後の並びで$i\,(0 \leq i \leq 11)$番目の接尾辞が$\boldsymbol{r}_i$である．また，$\boldsymbol{r}_i$の$\boldsymbol{x}$上の開始インデックスが$\mathsf{S}[i]$の値となる．

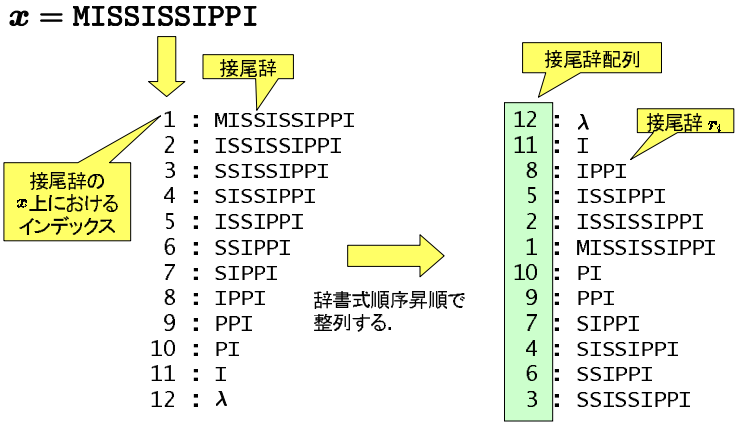


図 2.1: $\boldsymbol{x} = \mathtt{MISSISSIPPI}$のときの接尾辞配列

2.7 共通接頭辞の最大長

2つの任意の系列 $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{X}^*$ には，共通接頭辞 $\boldsymbol{u} \in \mathcal{P}(\boldsymbol{v}) \cap \mathcal{P}(\boldsymbol{w})$ がある．この共通接頭辞の最大の系列長を

$$\mathrm{lcp}\,(\boldsymbol{v}, \boldsymbol{w}) = \max\{|\boldsymbol{u}||\boldsymbol{u} \in \mathcal{P}(\boldsymbol{v}) \cap \mathcal{P}(\boldsymbol{w})\}$$

とする．記号 $\perp$ は $\mathcal{X}$ に属さないので，$\boldsymbol{v}$ との共通接頭辞は存在しない．そこで便宜上，$\mathrm{lcp}\,(\perp, \boldsymbol{v}) = \mathrm{lcp}\,(\boldsymbol{v}, \perp) = -1$ とする．

2.8 L配列

L配列は高さ配列 [11],[12] とも呼ばれ，2つの接尾辞 $\boldsymbol{r}_i, \boldsymbol{r}_{i+1}$ $(-1 \leq i \leq n)$ の最長共通接頭辞の系列長を配列で表現したものである．

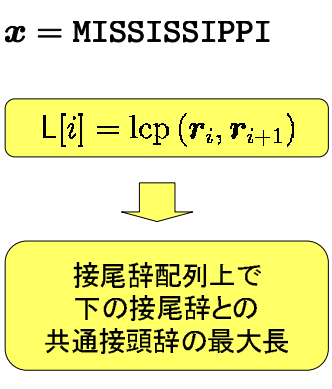
すなわち，$\boldsymbol{x}$ のL配列Lとは，Lの i 番目の要素が

$$\mathsf{L}[i] = \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_{i+1}) \tag{2.5}$$

で与えられる配列である．なお，$\mathsf{L}[-1] = \mathsf{L}[n] = -1$ で，Lの長さが $n+2$ であることを注記する．

例 6 図 2.2 に，$\boldsymbol{x} =$ MISSISSIPPI（系列長 11）のL配列の概要を示す．

$\boldsymbol{r}_3$ と $\boldsymbol{r}_4$ の最長共通接頭辞は ISSI で，系列長は4である．よって，$\mathsf{L}[3] = 4$ となる．便宜上，$\mathsf{L}[-1] = \mathsf{L}[n] = -1$ は境界を示すために存在する．



i	S[i]	$\boldsymbol{r}_i$	L[i]
-1		⊥	-1
0	12		0
1	11	I	1
2	8	IPPI	1
3	5	ISSIPPI	4
4	2	ISSISSIPPI	0
5	1	MISSISSIPPI	0
6	10	PI	1
7	9	PPI	0
8	7	SIPPI	2
9	4	SISSIPPI	1
10	6	SSIPPI	3
11	3	SSISSIPPI	-1
		⊥	

図 2.2: $\boldsymbol{x} =$ MISSISSIPPI のときのL配列

2.9 被覆集合

系列$\boldsymbol{x}$上の任意の部分系列$\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})$に対して，$\boldsymbol{u}$を接頭辞にもつ接尾辞が必ず存在する．そこで，

$$\mathcal{C}(\boldsymbol{u}) = \{i | \boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i), 0 \leq i \leq n\} \tag{2.6}$$

を定義し，**被覆集合**(cover set)と呼ぶことにする．接尾辞が整列しているから，$\mathcal{C}(\boldsymbol{u})$は連続した整数からなる集合である．すなわち，

$$\mathcal{C}(\boldsymbol{u}).\min = \min\{i | i \in \mathcal{C}(\boldsymbol{u})\}, \tag{2.7}$$

$$\mathcal{C}(\boldsymbol{u}).\max = \max\{i | i \in \mathcal{C}(\boldsymbol{u})\} \tag{2.8}$$

とすると，

$$\mathcal{C}(\boldsymbol{u}) = \{i | \mathcal{C}(\boldsymbol{u}).\min \leq i \leq \mathcal{C}(\boldsymbol{u}).\max\} \tag{2.9}$$

と表せる．なお，被覆集合は，文献[12]のrank intervalを整数の集合で表現したものである．ここで，$|\mathcal{C}(\boldsymbol{u})|$は$\boldsymbol{u}$が$\boldsymbol{x}$上に出現する数に等しく，また，$\mathcal{C}(\lambda) = \{0, 1, 2, \ldots, n\}$であることを注記する．

例 7 図2.3に，$\boldsymbol{x}$ = MISSISSIPPI（系列長11）の被覆集合の概要を示す．

部分系列Sの被覆集合を求めるとき，接尾辞の先頭部分にSがあるものを探す．この場合，接尾辞$\boldsymbol{r}_8, \boldsymbol{r}_9, \boldsymbol{r}_{10}, \boldsymbol{r}_{11}$が該当する．したがって，$\mathcal{C}(\mathrm{S}) = \{8, 9, 10, 11\}$となる．

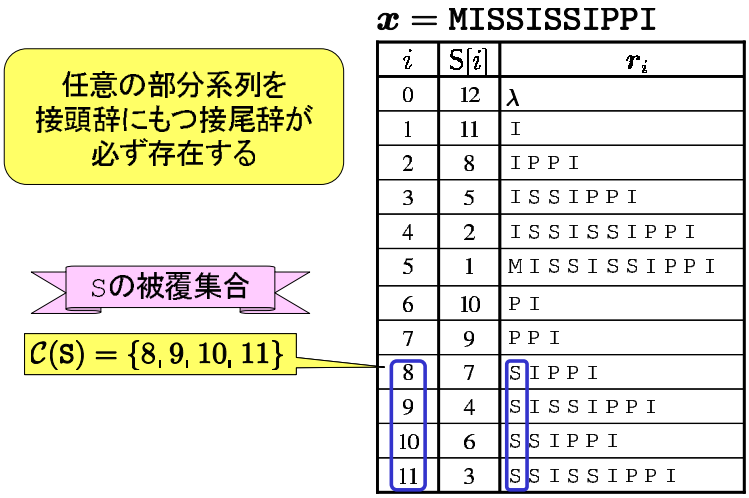


図 2.3: $\boldsymbol{x}$ = MISSISSIPPI のときの S の被覆集合

2.10 接頭記号と接尾記号

系列 $\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})$ と任意の記号 $\alpha \in \mathcal{X}$ に対して，$\boldsymbol{x}$ 上に $\alpha\boldsymbol{u}$ が出現するなら，α を $\boldsymbol{u}$ の**接頭記号**(head symbol) と呼ぶことにする．さらに，$\boldsymbol{u}$ の接頭記号の全ての集合を，$\boldsymbol{u}$ の**全接頭記号集合**と呼ぶことにし，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$ と表記する．すなわち，

$$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \{\alpha \in \mathcal{X} | \alpha\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})\} \tag{2.10}$$

である．

さらに，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$ は，接尾辞配列 S と被覆集合 $\mathcal{C}(\boldsymbol{u})$ を用いて

$$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \{x_i | i = \mathsf{S}[k] - 1, k \in \mathcal{C}(\boldsymbol{u}), i \geq 1\} \tag{2.11}$$

と表すこともできる．

また，$\boldsymbol{x}$ 上に $\boldsymbol{u}\beta$ が出現するなら，β を $\boldsymbol{u}$ の**接尾記号**(tail symbol) と呼ぶことにする．さらに，$\boldsymbol{u}$ の接尾記号の全ての集合を，$\boldsymbol{u}$ の**全接尾記号集合**と呼ぶことにし，$\mathcal{X}^{\mathrm{T}}(\boldsymbol{u})$ と表記する．すなわち，

$$\mathcal{X}^{\mathrm{T}}(\boldsymbol{u}) = \{\beta \in \mathcal{X} | \boldsymbol{u}\beta \in \mathcal{D}(\boldsymbol{x})\} \tag{2.12}$$

である．

なお，$\mathcal{X}^{\mathrm{H}}(\lambda)$ と $\mathcal{X}^{\mathrm{T}}(\lambda)$ は $\boldsymbol{x}$ 上に出現する全ての記号であることを注記する．また，$\boldsymbol{u}$ が $\boldsymbol{x}$ 上に 1 箇所のみ出現し，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{x})$ のときは，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \phi$ である．同じく，$\boldsymbol{u}$ が $\boldsymbol{x}$ 上に 1 箇所のみ出現し，$\boldsymbol{u} \in \mathcal{S}(\boldsymbol{x})$ のときは，$\mathcal{X}^{\mathrm{T}}(\boldsymbol{u}) = \phi$ である．

例 8 図2.4に，$\boldsymbol{x} = \mathtt{MISSISSIPPI}$ の接尾辞配列と接頭記号に関する概要を示す．

接尾辞 $\boldsymbol{r}_3 = \mathtt{ISSIPPI}$ の $\boldsymbol{x}$ 上における開始インデックスは $\mathsf{S}[3] = 5$ である．すなわち，記号 $x_{\mathsf{S}[3]-1} = \mathtt{S}$ は，$\boldsymbol{r}_3$ の接頭記号である．さらに，$\boldsymbol{r}_3$ の接頭辞についても，開始インデックスは変わらないので，記号 $x_{\mathsf{S}[3]-1}$ は，$\boldsymbol{r}_3$ の接頭辞の接頭記号である．

全接頭記号集合は，被覆集合を使うことで接尾辞配列から求めることができる．部分系列 S の被覆集合は，$\mathcal{C}(\mathsf{S}) = \{8, 9, 10, 11\}$ なので，$\boldsymbol{r}_8, \boldsymbol{r}_9, \boldsymbol{r}_{10}, \boldsymbol{r}_{11}$ の接頭記号を集めたものが全接頭記号集合 $\mathcal{X}^{\mathrm{H}}(\mathsf{S})$ になる．

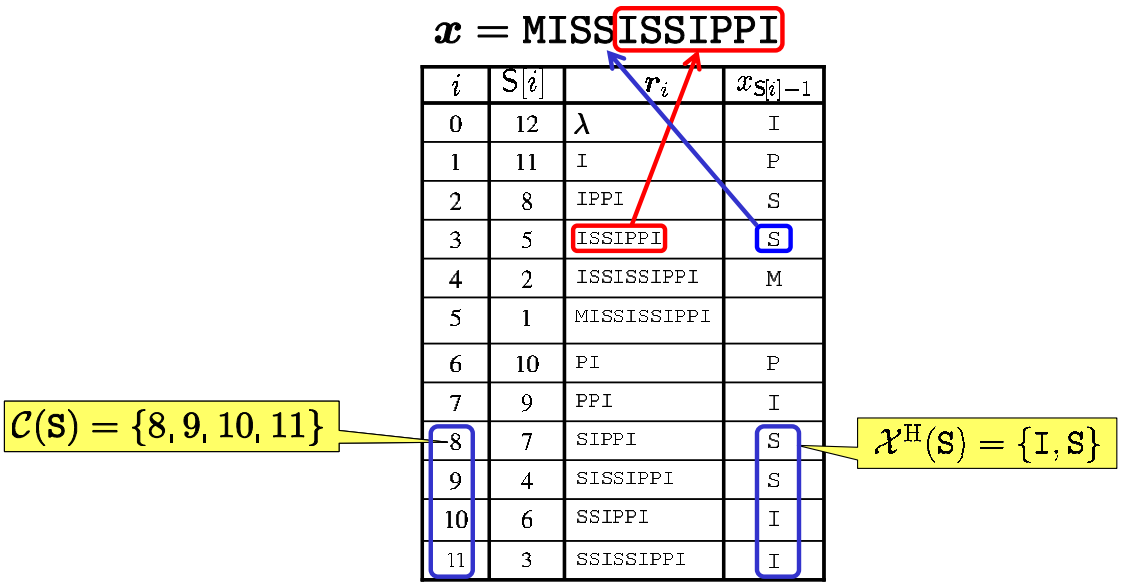


図 2.4: $\boldsymbol{x} =$ MISSISSIPPI のときの S の接尾辞配列と接頭記号

第3章　接尾辞木を用いた反辞書構築

3.1　反辞書構築のアプローチ

反辞書構築の従来手法では，入力系列から直接反辞書を構築するのではなく，一度接尾辞を用いたデータ構造を構築し，そこから反辞書を構築している．また，入力系列を全て入力してから反辞書を構築する静的構築と，入力系列を1記号ずつ読み込みながら逐次的に反辞書を変化させて構築する動的構築の2種類がある（図3.1）．

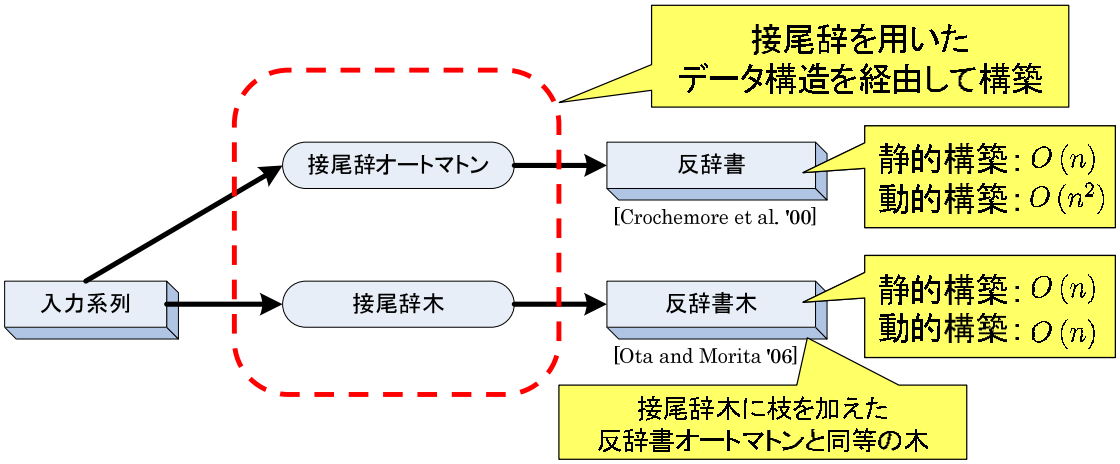


図 3.1: これまでの反辞書の構築方法

接尾辞を用いたデータ構造の多くは入力系列をすべて読み込み終わってから構築するアルゴリズムが多いため，反辞書を静的に構築する場合には，これらのデータ構造を活用できる利点がある．ただし，データ圧縮などに応用するときには，構築した反辞書を復号側に送るなどのオーバヘッドが生じる．

一方，反辞書を動的に構築する場合には，入力記号に合わせながら反辞書を変化させていくので，オーバーヘッドを無くすことができる．また，1記号読み込んだ後の反辞書の変化が少ない場合，第4章の提案手法よりも効率がよくなる可能性がある．

第 4 章と第 5 章では，静的構築と動的構築の両方で入力系列長に対して線形の計算量を達成している，太田・森田によって提案された接尾辞木を用いた反辞書構築手法 [9] を参考にして，新しい構築手法を提案した．

そこで，この章では接尾辞木を用いた反辞書構築手法について述べる．

3.2 接尾辞木

系列の集合を木構造で表現したものとして，トライ (trie)[13] がある．トライの枝にはラベルが付けられ，記号が 1 つ割り振られている．そして，与えられた系列 $\boldsymbol{x}$ の全ての接尾辞の集合 $\mathcal{S}(n)$ をトライで表現した木を接尾辞トライと呼ぶ．枝のラベルには，接尾辞上の記号が 1 つ割り振られている．

一方，接尾辞木 (Suffix tree)[14] も同様に，$\mathcal{S}(n)$ を木構造で表現したものである．接尾辞トライと異なるのは，ラベルには $\boldsymbol{x}$ の部分系列を割り振っていることである．ここで，接尾辞トライのノードのうち，子ノードを 1 つだけ持つノードを陰ノード，子ノードを 2 つ以上持つノードを陽ノードと呼ぶことにする．任意の陰ノードを p，p の親ノードを q, p の子ノードを r とし，p と q を接続する枝のラベルを α，p と r を接続する枝のラベルを β とする．このとき，系列 $\alpha\beta$ は $\boldsymbol{x}$ の部分系列である．したがって，枝に $\boldsymbol{x}$ の部分系列を割り振れるのならば，p をなくして，q と r を 1 本の枝（ラベルは $\alpha\beta$）にまとめることができる．このようにして，接尾辞トライの陰ノードを 1 本の枝にまとめ，接尾辞トライをコンパクトに表現したものが接尾辞木である．

ところで，接尾辞木の枝のラベルの長さが 2 以上の時，枝を記号単位で分割した区切り位置に仮想のノードがあると考えれば接尾辞トライと等価な木になる．このとき，この仮想ノードは接尾辞トライの陰ノードに対応し，接尾辞木の中間ノードは接尾辞トライの陽ノードに対応する．本論文では，接尾辞木において，上記の仮想ノードを扱うことがあるので，この仮想ノードを陰ノードと呼ぶことにする．

また，陽ノードから子ノードに向かって，枝が伸びている．これらの枝のラベルの先頭記号は，各々異なる記号である．そこで，陽ノードから子ノードへ記号 α を先頭に持つラベルの枝が伸びているとき，α が発芽すると呼ぶことにする．

図 3.2 に $\boldsymbol{x} = \mathtt{aababacbcb}$ の接尾辞トライと接尾辞木を示す．

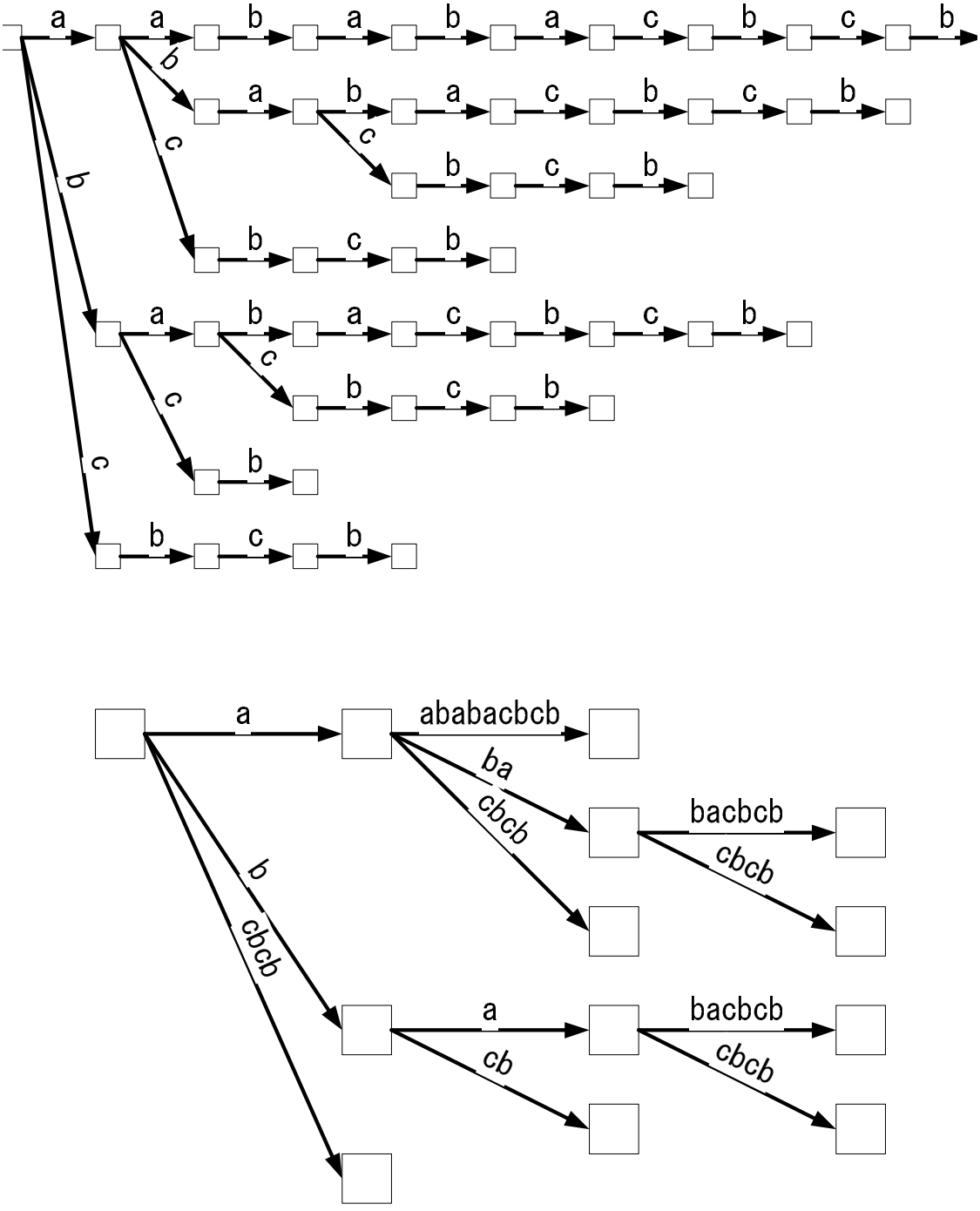


図 3.2: $\boldsymbol{x} =$ aababacbcb のときの接尾辞トライ (上) と接尾辞木 (下)

本論文では，$\boldsymbol{x}$ の接尾辞木を $\mathbb{T}$ と表記する．

また，任意の系列 $\boldsymbol{v} \in \mathcal{X}^*$ に対して，接尾辞木 $\mathbb{T}$ の根から葉ノードに向かって $\boldsymbol{v}$ と一致するようにラベルをたどることで到達したノードをローカスと呼び，$\mathcal{N}(\boldsymbol{v})$ と表記する．系列 $\boldsymbol{v}$ によっては，$\mathcal{N}(\boldsymbol{v})$ は陰ノードだったり，存在しないこともありえる．$\mathcal{N}(\boldsymbol{v})$ が存在するときは，$\boldsymbol{v} \in \mathcal{D}(\boldsymbol{x})$ である．

さらに，陰ノードを含む任意のノード p に対して，木の根から p までのラベルを順につないだ記号列をラベル列と呼び，$\boldsymbol{L}(p)$ と表記する．

3.3 suffix link と MF–link

接尾辞木を線形時間で構築する手法として，Ukkonen の構築手法 [15] がある．この手法では，接尾辞木の陽ノードに suffix link を用いている．suffix link とは，接尾辞木 $\mathbb{T}$ の陽ノード p に対して，$\sigma(\boldsymbol{L}(p))$ のラベル列を持つノードへのリンクである．suffix link 先のノードを $\mathcal{S}(p)$ と表記する．

また，太田らは，接尾辞木を用いて反辞書を作成するために，MF–link を定義した [9]．これは，ノード p の逆 suffix link である．すなわち，ノード m のラベル列 $\boldsymbol{L}(m)$ が $\sigma(\boldsymbol{L}(m)) = \boldsymbol{L}(p)$ を満たすとき，p から m へ MF–link が張られる．なお，m は陽ノードだけでなく陰ノードのものもあることを注記する．

図 3.3 に，$\boldsymbol{x} =$ `abbca` の接尾辞木と suffix link, MF–link の例を示す．白丸のノードが接尾辞木の陽ノード，小さな黒丸が陰ノード，四角が葉ノードである．suffix link は，実線の矢印で示している．MF–link は，点線矢印で示している．

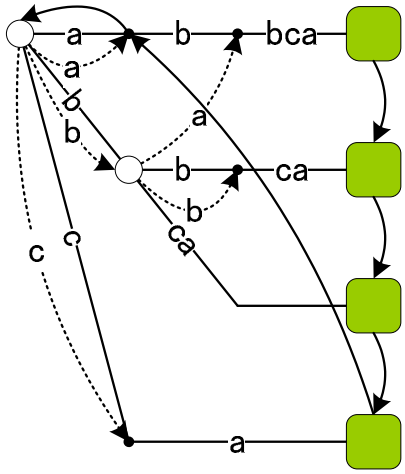


図 3.3: 接尾辞の suffix link（実線矢印）と MF–link（点線矢印）

3.4 反辞書の構築

接尾辞木を用いた反辞書構築手法では，極小禁止語をラベル列に持つMFWノードを接尾辞木に追加する．そして，MFWノードを追加する位置によって，極小禁止語を2種類に分けている．

- 葉ノードにMFWノードを追加する極小禁止語（タイプI）．
- 葉ノード以外にMFWノードを追加する極小禁止語（タイプII）．

Ukkonenの接尾辞木の構築手法では，1記号づつ読み込みながら接尾辞木を変化させている．1記号の読み込みにつき，葉ノードと陽ノードが高々1つ増える．したがって，接尾辞木の変化に追従しながら，上記の2種類の極小禁止語を求めていくことで，反辞書を構築する．

3.4.1 タイプIの極小禁止語の追加

タイプIの極小禁止語の場合，接尾辞木の葉ノードにMFWノードを追加する．すなわち，葉ノードをqとおいたとき，極小禁止語は$\boldsymbol{L}(q)\beta$ $(\beta \in \mathcal{X})$となる．このとき，$\boldsymbol{L}(q) \in \mathcal{D}(\boldsymbol{x})$ なので，極小禁止語の3つの条件のうち，(2.2)を満たしている．さらに，qは葉ノードなので，どんなβでも，$\boldsymbol{L}(q)\beta \notin \mathcal{D}(\boldsymbol{x})$となり，極小禁止語の3条件のうち，(2.1)を満たす．よって，(2.3)を満たすβを求めればよい．

ここで，βはqのsuffix linkを用いて求める．qのsuffix link先$\mathcal{S}(q)$のラベル列は$\sigma(\boldsymbol{L}(q))$である．すなわち，$\mathcal{S}(q)$から発芽する記号を$\beta$とすれば，$\sigma(\boldsymbol{L}(q))\beta \in \mathcal{D}(\boldsymbol{x})$となり，(2.3)を満たす．

こうして，ノードqから，求めたβを枝のラベルにしてMFWノードを伸ばして追加する．

なお，ラベル列が最短でない葉ノードのsuffix link先は，別の葉ノードである．したがって，suffix link先で発芽する記号はない．言い換えると，タイプIの極小禁止語のMFWノードは，ラベル列が最短の葉ノードだけに追加する．

例 9 図3.4に，$\boldsymbol{x} = \mathtt{abbca}$のタイプIのMFWノードを追加するときの様子を示す．図の白丸のノードが接尾辞木の陽ノード，四角のノードが接尾辞木の葉ノードである．三角のノードが追加するMFWノードである．

ラベル列が最短の葉ノードをqとすると，$\boldsymbol{L}(q) = \mathtt{ca}$である．すなわち，タイプIの極小禁止語は$\mathtt{ca}\beta$ $(\beta \in \mathcal{X})$とおける．

ここで，q の suffix link 先 $\mathcal{S}(q)$ のラベル列は，a である．また，$\mathcal{S}(q)$ からは，b が発芽しているので，ab $\in \mathcal{D}(\boldsymbol{x})$ である．よって，β が b のときは，cab が極小禁止語の 3 条件を満たすので，葉ノード q から MFW ノードを伸ばしている．

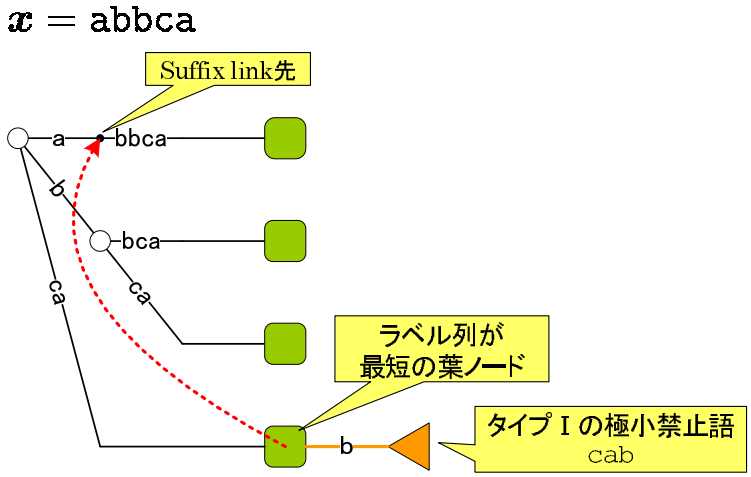


図 3.4: 接尾辞木へのタイプ I の反辞書の追加例

3.4.2 タイプ II の極小禁止語の追加

タイプ II の極小禁止語の場合，接尾辞木の葉ノード以外のノード，つまり陽ノードや陰ノードに MFW ノードを追加する．追加する位置は，陽ノード p から張られている MF–link 先 m である．すなわち，極小禁止語は $\boldsymbol{L}(m)\beta$ となる．このとき，$\boldsymbol{L}(m) \in \mathcal{D}(\boldsymbol{x})$, なので，極小禁止語の 3 つの条件のうち，(2.2) を満たしている．よって，(2.1) と (2.3) を満たす β を求めればよい．

ここで，p と m から発芽する記号に注目する．p は陽ノードなので，複数の記号が発芽している．そこで，p で発芽し，m で発芽しない記号を β とすれば，$\boldsymbol{L}(m)\beta \notin \mathcal{D}(\boldsymbol{x}), \sigma(\boldsymbol{L}(m)\beta) = \boldsymbol{L}(p)\beta \in \mathcal{D}(\boldsymbol{x})$ となり，(2.1) と (2.3) を満たす．

こうして，ノード m から，求めた β を枝のラベルにして MFW ノードを伸ばして追加する．

例 10 図 3.5 に，$\boldsymbol{x}$ = abbca のタイプ II の MFW ノードを追加するときの様子を示す．

ラベル列が b の陽ノードを p，ノード p の MF–link 先のうち，ラベル列が ab のノードを m とする．このとき，タイプ II の極小禁止語は abβ ($\beta \in \mathcal{X}$) とおける．

ノード p から発芽している記号は b と c，ノード m から発芽している記号は b である．よって，β が c のときは，abc が極小禁止語の 3 条件を満たすので，ノード m から MFW ノードを伸ばしている．

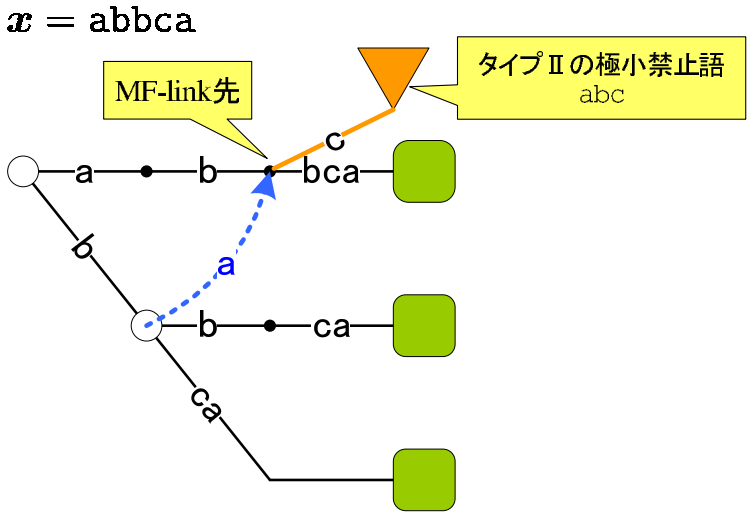


図 3.5: 接尾辞木へのタイプ II の反辞書の追加例

第4章　配列を用いた静的反辞書構築の提案

4.1　接尾辞木を用いた反辞書構築手法の記憶量の内訳

接尾辞木を用いた反辞書構築手法は，計算量が線形とはいえ，アルファベットが多値の場合には係数が大きくなるという問題がある．この問題を解決するために，使用している記憶量の内訳を調べると，記憶量はMF–linkを保持するためにほとんど占められていることがわかった．これは，MF–linkが使用する記憶量がアルファベットサイズにも比例するのが原因である．

この問題を解決するためには，MF–linkを保持しないで反辞書を構築する新たな手法が必要となる．

そこで本研究では，接尾辞木よりも少ない記憶量で接尾辞を表現できる接尾辞配列に注目した．この章では，接尾辞配列を用いて，MF–linkを保持しない新たな反辞書構築手法を提案する．

4.2　反辞書の別表現

接尾辞配列を用いて極小禁止語を求めるために，極小禁止語を先頭記号αと系列$\boldsymbol{u}$の2つに分割することを提案する [1], [2].

そのために，(2.10)で定義した全接頭記号集合$\mathcal{X}^{\mathrm{H}}(\cdot)$を用いて，$\mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{u}))$と$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$との差集合を$\mathcal{H}(\boldsymbol{u})$と表記し，

$$\mathcal{H}(\boldsymbol{u}) = \begin{cases} \mathcal{X} \setminus \mathcal{X}^{\mathrm{H}}(\lambda) & (\boldsymbol{u} = \lambda) \\ \mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{u})) \setminus \mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) & (\boldsymbol{u} \neq \lambda) \end{cases} \tag{4.1}$$

と定義する．

そして，$\mathcal{H}(\boldsymbol{u})$を用いて反辞書$\mathcal{A}(\boldsymbol{x})$を表せることを，次の定理1で示す．

定理 1 任意の$\boldsymbol{x} \in \mathcal{X}^*$に対して，

$$\mathcal{A}(\boldsymbol{x}) = \{\alpha\boldsymbol{u} | \alpha \in \mathcal{H}(\boldsymbol{u}), \boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})\}.$$

証明 最初に，$\alpha\boldsymbol{u} \in \mathcal{A}(\boldsymbol{x})$ $(\alpha \in \mathcal{X}, \boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}))$ とおいたとき，$\alpha \in \mathcal{H}(\boldsymbol{u})$ であることを，$\boldsymbol{u} = \lambda$ と $\boldsymbol{u} \neq \lambda$ の場合に分けて証明する．

まず，$\boldsymbol{u} = \lambda$ の場合，$\alpha \in \mathcal{A}(\boldsymbol{x})$ となるので，α は $\boldsymbol{x}$ 上に出現していない記号である．ここで，$\mathcal{X}^{\mathrm{H}}(\lambda)$ は $\boldsymbol{x}$ 上に出現する全ての記号の集合なので，$\alpha \in \mathcal{X} \setminus \mathcal{X}^{\mathrm{H}}(\lambda) = \mathcal{H}(\boldsymbol{u})$ である．

また，$\boldsymbol{u} \neq \lambda$ の場合，$\alpha\boldsymbol{u} \in \mathcal{A}(\boldsymbol{x})$ から，$\alpha\boldsymbol{u} \notin \mathcal{D}(\boldsymbol{x}), \alpha\pi(\boldsymbol{u}) \in \mathcal{D}(\boldsymbol{x})$ となる．(2.10) から，$\alpha \notin \mathcal{X}^{\mathrm{H}}(\boldsymbol{u}), \alpha \in \mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{u}))$ となる．よって，$\alpha \in \mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{u})) \setminus \mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \mathcal{H}(\boldsymbol{u})$ である．

次に，$\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}), \alpha \in \mathcal{H}(\boldsymbol{u})$ ならば，$\alpha\boldsymbol{u} \in \mathcal{A}(\boldsymbol{x})$ であることを，$\boldsymbol{u} = \lambda$ と $\boldsymbol{u} \neq \lambda$ の場合に分けて証明する．

まず，$\boldsymbol{u} = \lambda$ の場合，$\mathcal{X}^{\mathrm{H}}(\lambda)$ は $\boldsymbol{x}$ 上に出現する全ての記号の集合である．よって，(4.1) より，α は未出現記号である．このとき，$\alpha\boldsymbol{u}$ は，(2.1), (2.2), (2.3) を満す．よって，$\alpha\boldsymbol{u} \in \mathcal{A}(\boldsymbol{x})$ である．

また，$\boldsymbol{u} \neq \lambda$ の場合，(4.1) から，$\alpha \in \mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{u})), \alpha \notin \mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$ である．このとき (2.10) から，$\alpha\pi(\boldsymbol{u}) \in \mathcal{D}(\boldsymbol{x}), \alpha\boldsymbol{u} \notin \mathcal{D}(\boldsymbol{x})$ となり，(2.1) と (2.2) を満す．また，$\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})$ なので，(2.3) を満す．よって，$\alpha\boldsymbol{u} \in \mathcal{A}(\boldsymbol{x})$ である．

以上より，題意が成り立つ．

□

ところで，(2.12) で定義した全接尾記号集合 $\mathcal{X}^{\mathrm{T}}(\cdot)$ を用いて，

$$\mathcal{X}^{\mathrm{T}}(\boldsymbol{u}) = \{\beta \in \mathcal{X} | \boldsymbol{u}\beta \in \mathcal{D}(\boldsymbol{x})\}, \tag{4.2}$$

$$\mathcal{T}(\boldsymbol{u}) = \begin{cases} \mathcal{X} \setminus \mathcal{X}^{\mathrm{T}}(\lambda) & (\boldsymbol{u} = \lambda) \\ \mathcal{X}^{\mathrm{T}}(\sigma(\boldsymbol{u})) \setminus \mathcal{X}^{\mathrm{T}}(\boldsymbol{u}) & (\boldsymbol{u} \neq \lambda) \end{cases} \tag{4.3}$$

と定義すると，定理 1 と同様に次の系 1 が成り立つ．

系 1 任意の $\boldsymbol{x} \in \mathcal{X}^*$ に対して，

$$\mathcal{A}(\boldsymbol{x}) = \{\boldsymbol{u}\beta | \beta \in \mathcal{T}(\boldsymbol{u}), \boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})\}.$$

なお，系 1 は，この章では使わないが，第 5 章で用いる．

4.3 極小禁止語の先頭文字を削除した系列

任意の部分系列 $\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})$ が $\mathcal{H}(\boldsymbol{u}) \neq \phi$ のとき，

$$\mathcal{G}_{\mathrm{H}} = \{\boldsymbol{u} | \boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}), \mathcal{H}(\boldsymbol{u}) \neq \phi\} \tag{4.4}$$

と定義する．そして，

$$\sigma(\mathcal{A}(\boldsymbol{x})) = \{\sigma(\boldsymbol{w}) | \boldsymbol{w} \in \mathcal{A}(\boldsymbol{x})\}$$

とおくと，定理 1 より，

$$\mathcal{G}_{\mathrm{H}} = \sigma(\mathcal{A}(\boldsymbol{x}))$$

が成立する．

この $\mathcal{G}_{\mathrm{H}}$ を求めることができれば，定理 1 から，要素 $\boldsymbol{u} \in \mathcal{G}_{\mathrm{H}}$ に対して $\mathcal{H}(\boldsymbol{u})$ を求めることで反辞書を構築できる．しかし，$\mathcal{G}_{\mathrm{H}}$ を求めることは容易ではない．そこで，容易に求めれて，$\mathcal{G}_{\mathrm{H}}$ を含む集合を考えることにする．

まずは，(2.4) で定義したランク i の接尾辞 $\boldsymbol{r}_i$ $(0 \leq i \leq n)$ ごとに，$\boldsymbol{r}_i$ の全ての接頭辞の集合 $\mathcal{P}(\boldsymbol{r}_i)$ を 2 つに分割した集合 $\mathcal{F}(i), \mathcal{E}(i)$ を定義する．

$$\mathcal{F}(i) \ = \ \{\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i) \,||\boldsymbol{u}| \neq \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_k) + 1, -1 \leq {}^{\forall}k \leq n+1\}, \tag{4.5}$$

$$\mathcal{E}(i) \ = \ \{\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i) \,||\boldsymbol{u}| = \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_k) + 1, -1 \leq {}^{\exists}k \leq n+1\}. \tag{4.6}$$

明らかに，$\mathcal{P}(\boldsymbol{r}_j) = \mathcal{F}(j) \cup \mathcal{E}(j), \mathcal{F}(j) \cap \mathcal{E}(j) = \phi$ となる．

次の補題 1 で，$\mathcal{F}(i)$ と $\mathcal{G}_{\mathrm{H}}$ の関係を示す.

補題 1 任意の $\boldsymbol{v} \in \mathcal{X}^*$ とある i $(0 \leq i \leq n)$ に対して，$\boldsymbol{v} \in \mathcal{F}(i)$ ならば，

$$\mathcal{H}(\boldsymbol{v}) = \phi.$$

証明 空系列 λ は，$\lambda \in \mathcal{E}(i)$ である．よって，$\boldsymbol{v} \neq \lambda$. そこで，$\boldsymbol{v} = \boldsymbol{u}\beta$ $(\boldsymbol{u} = \pi(\boldsymbol{v}), \beta \in \mathcal{X})$ とおく．

ここで，背理法を使って証明する．いま，$\mathcal{H}(\boldsymbol{v}) \neq \phi$ と仮定する．このとき，ある $\alpha \in \mathcal{H}(\boldsymbol{v})$ が存在し，$\alpha \in \mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) \setminus \mathcal{X}^{\mathrm{H}}(\boldsymbol{u}\beta)$ とおける．すなわち，$\alpha\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}), \alpha\boldsymbol{u}\beta \notin \mathcal{D}(\boldsymbol{x})$ である．したがって，ある $\boldsymbol{w} \in \mathcal{X}^*$ が存在し，$\alpha\boldsymbol{u}\boldsymbol{w} \in \mathcal{D}(\boldsymbol{x}), \beta \notin \mathcal{P}(\boldsymbol{w})$ とおける．このとき，$\mathrm{lcp}\,(\boldsymbol{u}\beta, \boldsymbol{u}\boldsymbol{w}) = |\boldsymbol{u}|$ である．したがって，$\boldsymbol{u}\beta \in \mathcal{P}(\boldsymbol{r}_i)$ だから，$\mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{u}\boldsymbol{w}) = |\boldsymbol{u}|$ となる．

一方，$\boldsymbol{u}\boldsymbol{w} \in \mathcal{D}(\boldsymbol{x})$ である．よって，ある整数 k が存在し，$\boldsymbol{u}\boldsymbol{w} \in \mathcal{P}(\boldsymbol{r}_k)$ とおける．このとき，$\mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_k) = |\boldsymbol{u}|$ となる．

しかしながら，$|\boldsymbol{v}| = |\boldsymbol{u}| + 1$ なので，これは $|\boldsymbol{v}| \neq \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_k) + 1$ という仮定に矛盾する．よって，$\mathcal{H}(\boldsymbol{v}) = \phi$ である．

□

補題 1 から $\mathcal{F}(i) \cap \mathcal{G}_{\mathrm{H}} = \phi$ である．すなわち，$\mathcal{G}_{\mathrm{H}}$ の要素 $\boldsymbol{u}$ は $\mathcal{E}(i)$ から探せばよい．ここで，$\boldsymbol{u}$ を効率よく探すため，$\mathcal{E}(i)$ に含まれる 2 つの系列に注目する．すなわち，(2.5) で定義した L 配列を用いて，$\boldsymbol{r}_i$ ごとに，以下の条件を満たす $\boldsymbol{r}_i$ の接頭辞 $\bar{\boldsymbol{t}}_i, \bar{\boldsymbol{b}}_i \in \mathcal{P}(\boldsymbol{r}_i)$ を定義する．

$$|\bar{\boldsymbol{t}}_i| = \mathsf{L}[i-1] + 1, \tag{4.7}$$

$$|\bar{\boldsymbol{b}}_i| = \mathsf{L}[i] + 1 \leq |\boldsymbol{r}_i|. \tag{4.8}$$

なお，$\mathsf{L}[i] = |\boldsymbol{r}_i|$ のとき，$\bar{\boldsymbol{b}}_i$ は存在しないことを注記する．

例として，$\boldsymbol{x} = \mathtt{ababb}$ のときの $\bar{\boldsymbol{t}}_i, \bar{\boldsymbol{b}}_i$ を表 4.1 に示す．

表 4.1: $\boldsymbol{x} = \mathtt{ababb}$ のときの $\bar{\boldsymbol{t}}_i$ と $\bar{\boldsymbol{b}}_i$（$\bar{\boldsymbol{b}}_i$ の—は，存在しない事を示す）．

i	$\boldsymbol{r}_i$	$\mathsf{L}[i]$	$\bar{\boldsymbol{t}}_i$	$\bar{\boldsymbol{b}}_i$
-1	$\perp$	-1		
0	λ	0	λ	—
1	ababb	2	a	aba
2	abb	0	abb	a
3	b	1	b	—
4	babb	1	ba	ba
5	bb	-1	bb	λ
6	$\perp$			

そして，$\bar{\boldsymbol{t}}_i$ と $\bar{\boldsymbol{b}}_i$ を要素に持つ集合 $\mathcal{G}(i)$ を定義する．

$$\mathcal{G}(i) = \begin{cases} \{\bar{\boldsymbol{t}}_i, \bar{\boldsymbol{b}}_i\} & (if\ \bar{\boldsymbol{b}}_i\ exists) \\ \{\bar{\boldsymbol{t}}_i\} & (otherwise) \end{cases} \tag{4.9}$$

次の補題 2 で，$\mathcal{E}(i)$ と $\mathcal{G}(i)$ の関係を示す．

補題 2 任意の $\boldsymbol{v} \in \mathcal{X}^*$ とある i $(0 \leq i \leq n)$ に対して，$\boldsymbol{v} \in \mathcal{E}(i)$ ならば，ある k が存在し，

$$\boldsymbol{v} \in \mathcal{G}(k)\,.$$

証明 最初に，(2.7),(2.8) で定義した被覆集合の最小値と最大値を用いて，$p = \mathcal{C}(\boldsymbol{v}).\min, q = \mathcal{C}(\boldsymbol{v}).\max$ とおく．

接尾辞配列が整列していることと，(4.6) から，$|\boldsymbol{v}| = \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_{p-1}) + 1$ または $|\boldsymbol{v}| = \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_{q+1}) + 1$ である．ここで，(2.6) から，$\mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{r}_i) = \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{r}_p)\,, \mathrm{lcp}\,(\boldsymbol{r}_i, \boldsymbol{r}_{q+1}) = \mathrm{lcp}\,(\boldsymbol{r}_q, \boldsymbol{r}_{q+1})$ である．

L配列から，$\mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{r}_p)+1 = \mathsf{L}[p-1]+1, \mathrm{lcp}\,(\boldsymbol{r}_q, \boldsymbol{r}_{q+1})+1 = \mathsf{L}[q]+1$ である．したがって，$|\boldsymbol{v}| = \mathsf{L}[p-1]+1$ または $|\boldsymbol{v}| = \mathsf{L}[q]+1$ である．すなわち，$\boldsymbol{v} = \bar{\boldsymbol{t}}_p$ または $\boldsymbol{v} = \bar{\boldsymbol{b}}_q$ となる．よって，(4.9) から，$\boldsymbol{v} \in \mathcal{G}(p)$ または $\boldsymbol{v} \in \mathcal{G}(q)$ である．

以上より，$\boldsymbol{v} \in \mathcal{G}(k)$ である．

□

ここで，全ての $\mathcal{G}(i)$ の和集合 $\mathcal{G}$ を定義する．

$$\mathcal{G} = \bigcup_{0 \le i \le n} \mathcal{G}(i)\,. \tag{4.10}$$

このとき，$\mathcal{G}_{\mathrm{H}}$ は $\mathcal{G}$ に含まれることを，次の定理 2 で示す．

定理 2 任意の $\boldsymbol{x} \in \mathcal{X}^*$ に対して，

$$\mathcal{G}_{\mathrm{H}} \subset \mathcal{G}.$$

証明 最初に，$\boldsymbol{v} \in \mathcal{G}_{\mathrm{H}}$ とおくと，(4.4) から，$\mathcal{H}(\boldsymbol{v}) \neq \phi$ である．よって，補題 1 の対偶から，任意の i $(0 \le i \le n)$ について，$\boldsymbol{v} \notin \mathcal{F}(i)$ となる．すなわち，

$$\boldsymbol{v} \notin \bigcup_{0 \le i \le n} \mathcal{F}(i)\,. \tag{4.11}$$

また，$j \in \mathcal{C}(\boldsymbol{v})$ とおくと，(2.6) から，$\boldsymbol{v} \in \mathcal{P}(\boldsymbol{r}_j)$. さらに，(4.11) から，$\boldsymbol{v} \notin \mathcal{F}(j)$. よって，$\boldsymbol{v} \in \mathcal{E}(j)$ となる．このとき，補題 2 から，ある k が存在して，$\boldsymbol{v} \in \mathcal{G}(k)$ である．したがって，(4.10) から，$\mathcal{G}_{\mathrm{H}} \subset \mathcal{G}$ である．

□

定理 2 から，$\mathcal{G}_{\mathrm{H}}$ は $\mathcal{G}$ に含まれる．さらに，(4.7),(4.8) から，$\mathcal{G}$ は接尾辞配列と L 配列を走査するだけで容易に求めることができる．

4.4 提案手法の概要

提案手法は，長さ n の系列 $\boldsymbol{x}$ を入力し，反辞書 $\mathcal{A}(\boldsymbol{x})$ を出力する．

反辞書は，$\mathcal{G}_{\mathrm{H}}$ と定理 1 を用いて構築する．ただし，$\mathcal{G}_{\mathrm{H}}$ を求めるのは容易でないことから，代わりに $\mathcal{G}$ を求める．定理 2 から，$\mathcal{G}_{\mathrm{H}}$ は $\mathcal{G}$ に含まれている．なお，(4.10) より $|\mathcal{G}|$ は高々 $2(n+1)$ であることから，全ての $\boldsymbol{u} \in \mathcal{G}$ について $\mathcal{G}_{\mathrm{H}}$ の要素かどうかを調べる．判別方法は，(4.4) から，$\mathcal{H}(\boldsymbol{u})$ が空集合かどうかを調べる．この判別方法だと，$\mathcal{H}(\boldsymbol{u})$ が空集合でないときに，極小禁止語を容易に求めることができる．すなわち，定理 1 から，$\mathcal{H}(\boldsymbol{u})$ の要素を $\boldsymbol{u}$ の先頭に加えると極小禁止語になる．

提案手法SA2ADの各ステップを以下に示す.

Algorithm SA2AD
　input　：入力記号系列$\boldsymbol{x}$
　output　：反辞書$\mathcal{A}(\boldsymbol{x})$
begin
　/*Step 1：**接尾辞配列の構築***/
　$\mathsf{S} \leftarrow$ construct_suffix_array$(\boldsymbol{x})$;
　/*Step 2：L**配列の構築***/
　$\mathsf{L} \leftarrow$ construct_l_array$(\boldsymbol{x}, \mathsf{S})$;
　/*Step 3：$\mathcal{G}$**の構築***/
　$\mathcal{G} \leftarrow$ construct_G(S, L) ;
　/*Step 4：**反辞書**$\mathcal{A}(\boldsymbol{x})$**の構築***/
　$\mathcal{A}(\boldsymbol{x}) \leftarrow$ construct_antidictionary$(\boldsymbol{x}, \mathsf{S}, \mathcal{G})$;
　return　$\mathcal{A}(\boldsymbol{x})$;
end.

Step 1とStep 2では，$\mathcal{G}$を構築する準備として接尾辞配列SとL配列Lを構築する．Step 3で$\mathcal{G}$を構築し，Step 4で$\mathcal{H}(\boldsymbol{u})$を求めて$\mathcal{A}(\boldsymbol{x})$を構築する．

Step 1のconstruct_suffix_array$(\boldsymbol{x})$は，計算量が線形の既知のアルゴリズム[16] – [20]を用いる．Step 2のconstruct_l_array$(\boldsymbol{x}, \mathsf{S})$も，計算量が線形の既知のアルゴリズム[11], [12]を用いる．Step 3のconstruct_G(S, L)の処理内容については4.6節で，Step 4のconstruct_antidictionary$(\boldsymbol{x}, \mathsf{S}, \mathcal{G})$については4.7節で述べる．

4.5　系列の表現

系列$\boldsymbol{u} \in \mathcal{G}$の$\mathcal{H}(\boldsymbol{u})$を求めるには，(4.1)より，(2.10)で定義した$\boldsymbol{u}$の全接頭記号集合$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$を求める必要がある．

ここで，(2.7),(2.8)で定義した$\boldsymbol{u}$の被覆集合の最小値$\mathcal{C}(\boldsymbol{u})$.minと最大値$\mathcal{C}(\boldsymbol{u})$.maxを用いると，(2.11)より，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$は接尾辞配列Sの$\mathcal{C}(\boldsymbol{u})$.min番目から$\mathcal{C}(\boldsymbol{u})$.max番目を走査して求めることができる．そこで，$\boldsymbol{u}$は3つの組

$$(|\boldsymbol{u}|, \mathcal{C}(\boldsymbol{u}).\min, \mathcal{C}(\boldsymbol{u}).\max) \tag{4.12}$$

で表現する．

ここで，$\mathcal{C}(\boldsymbol{u})$.min, $\mathcal{C}(\boldsymbol{u})$.maxとL配列との関係を，次の命題1で示す．

命題 1 系列 $\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x})$ に対して，$p = \mathcal{C}(\boldsymbol{u}).\min, q = \mathcal{C}(\boldsymbol{u}).\max$ とおくと，

$$|\boldsymbol{u}| > \mathsf{L}[p-1]\,, |\boldsymbol{u}| > \mathsf{L}[q]\,.$$

さらに，$p < q$ ならば，

$$|\boldsymbol{u}| \leq \min\{\mathsf{L}[i]\,|p \leq i \leq q-1\}\,.$$

証明 まず，p のとりうる値 $0 \leq p \leq n$ を，$p = 0$ と $1 \leq p \leq n$ に分けて考える．

はじめに，$p = 0$ のときを考える．(2.5) より $\mathsf{L}[p-1] = \mathsf{L}[-1] = -1$．一方，$|\boldsymbol{u}| \geq 0$ である．したがって，$|\boldsymbol{u}| > \mathsf{L}[p-1]$ である．

次に，$1 \leq p \leq n$ のときを考える．(2.7) より，$p-1 \notin \mathcal{C}(\boldsymbol{u})$．すなわち，(2.6) から，$\boldsymbol{u} \notin \mathcal{P}(\boldsymbol{r}_{p-1})$．よって，$|\boldsymbol{u}| > \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{u})$ である．このとき，任意の $\boldsymbol{v} \in \mathcal{X}^*$ に対して，$\mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{u}\boldsymbol{v}) = \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{u})$ となる．ところで，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_p)$ なので，$\boldsymbol{r}_p = \boldsymbol{u}\boldsymbol{w}$ $(\boldsymbol{w} \in \mathcal{X}^*)$ と表すと，$\mathsf{L}[p-1] = \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{r}_p) = \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{u}\boldsymbol{w}) = \mathrm{lcp}\,(\boldsymbol{r}_{p-1}, \boldsymbol{u})$ である．したがって，$|\boldsymbol{u}| > \mathsf{L}[p-1]$ である．

また，q のとりうる値 $0 \leq q \leq n$ を，$q = n$ と $0 \leq q \leq n-1$ に分けて考えると，上と同様の議論から $|\boldsymbol{u}| > \mathsf{L}[q]$ が示せる．

さらに，$p < q$ のときを考える．このとき，$k\,(p \leq k \leq q)$ に対して，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_k)$ である．したがって，$i\,(p \leq i \leq q-1)$ に対して，$|\boldsymbol{u}| \leq \min\{\mathsf{L}[i]\,|p \leq i \leq q-1\}$ である．

以上より，題意が成り立つ．

□

集合 $\mathcal{G}$ の要素のうち，(4.7) で定義した $\bar{\boldsymbol{t}}_i$ $(0 \leq i \leq n)$ の長さは定まっているので，次の系 2 が成り立つ．

系 2 ある $i\,(0 \leq i \leq n)$ が与えられたとき，系列 $\bar{\boldsymbol{t}}_i \in \mathcal{G}$ に対して，

$$\begin{aligned} \mathcal{C}(\bar{\boldsymbol{t}}_i).\min &= i, \\ \mathcal{C}(\bar{\boldsymbol{t}}_i).\max &= \min\{j||\bar{\boldsymbol{t}}_i| > \mathsf{L}[j]\,, i \leq j \leq n\}\,. \end{aligned}$$

同じく，集合 $\mathcal{G}$ の要素のうち，(4.8) で定義した $\bar{\boldsymbol{b}}_i \in \mathcal{G}$ の長さも定まっているので，次の系 3 が成り立つ．

系 3 ある $i\,(0 \leq i \leq n)$ が与えられたとき，系列 $\bar{\boldsymbol{b}}_i \in \mathcal{G}$ に対して，

$$\begin{aligned} \mathcal{C}\left(\bar{\boldsymbol{b}}_i\right).\min &= \max\left\{j||\bar{\boldsymbol{b}}_i| > \mathsf{L}[j-1]\,, 0 \leq j \leq i\right\}, \\ \mathcal{C}\left(\bar{\boldsymbol{b}}_i\right).\max &= i. \end{aligned}$$

提案手法では，$\bar{\boldsymbol{t}}_i, \bar{\boldsymbol{b}}_i$ は (4.12) の 3 つ組で表現する．なお，(4.7), (4.8) と系 2, 系 3 から，3 つ組の値は i と L だけで求めることができる．特に，$\mathcal{C}(\bar{\boldsymbol{t}}_i)$.max と $\mathcal{C}\big(\bar{\boldsymbol{b}}_i\big)$.min 以外は L を走査することなく決定できる．

4.6 Step 3 のアルゴリズム

Step 3 では，接尾辞配列 S と L 配列 L から，$\mathcal{G}$ を構築する．

集合 $\mathcal{G}$ の要素 $\bar{\boldsymbol{t}}_i, \bar{\boldsymbol{b}}_i$ は (4.12) の 3 つ組で表現するが，系 2 と系 3 から，$\mathcal{C}(\cdot)$.min と $\mathcal{C}(\cdot)$.max の求め方が $\bar{\boldsymbol{t}}_i$ と $\bar{\boldsymbol{b}}_i$ とで違う．そこで，$\mathcal{G}$ は 2 つの集合

$$\mathcal{G}_{\mathrm{T}} = \bigcup_{0\le i\le n} \{\bar{\boldsymbol{t}}_i\}, \tag{4.13}$$

$$\mathcal{G}_{\mathrm{B}} = \bigcup_{0\le i\le n} \{\bar{\boldsymbol{b}}_i\}. \tag{4.14}$$

を求めた後に，和集合を演算して構築する．なお，$\bar{\boldsymbol{b}}_i$ が存在しない時は，$\{\bar{\boldsymbol{b}}_i\} = \phi$ とする．

さらに，$\mathcal{C}(\cdot)$.min, $\mathcal{C}(\cdot)$.max を求めるときには，(2.11) より，(2.10) で定義した $\mathcal{X}^{\mathrm{H}}(\cdot)$ を用いて S を走査するが，$\mathcal{G}$ の要素が整列していると S の走査を抑えることができる．そこで，$\mathcal{G}_{\mathrm{T}}$ と $\mathcal{G}_{\mathrm{B}}$ が辞書式順序昇順で整列するように構築する．整列する手法については，この節の最後に述べる．また，$\mathcal{G}$ は，$\mathcal{G}_{\mathrm{T}}$ と $\mathcal{G}_{\mathrm{B}}$ のマージソート [21] で構築する．

そこで，辞書式順序昇順で整列した後の $\mathcal{G}_{\mathrm{T}}, \mathcal{G}_{\mathrm{B}}, \mathcal{G}$ を

$$\mathcal{G}_{\mathrm{T}} = \{\boldsymbol{t}_0, \ldots, \boldsymbol{t}_n\}\ (\boldsymbol{t}_i \prec \boldsymbol{t}_j, 0\le i<j\le n), \tag{4.15}$$

$$\mathcal{G}_{\mathrm{B}} = \{\boldsymbol{b}_0, \ldots, \boldsymbol{b}_k\}\ (\boldsymbol{b}_i \prec \boldsymbol{b}_j, 0\le i<j\le k) \tag{4.16}$$

$$\mathcal{G} = \{\boldsymbol{g}_0, \ldots, \boldsymbol{g}_m\}\ (\boldsymbol{g}_i \prec \boldsymbol{g}_j, 0\le i<j\le m), \tag{4.17}$$

とあらためて定義する．なお，$\bar{\boldsymbol{b}}_i$ が存在しないことがあるので，$k = |\mathcal{G}_{\mathrm{B}}| - 1 \le n$ である．$\mathcal{G}_{\mathrm{T}}$ と $\mathcal{G}_{\mathrm{B}}$ が同じ要素を持つことがあるので，$m = |\mathcal{G}| - 1 \le 2n + 1$ である．

Step 3 のアルゴリズム construct_G を以下に示す．アルゴリズム中の stack は，S のインデックスを格納するスタックである．stack.push(·) でインデックスを格納し，stack.pop() で取り出す．stack.is_empty() でスタックが空かどうかを調べ，空なら **true** を返し，空でなければ **false** を返す．

Algorithm construct_G

input ：接尾辞配列 S, L 配列 L

```
    output  : 系列の集合 𝒢
begin                                                                    1
  /* Step 3–1 : 初期設定 */                                              2
  j ← n                                                                  3
  for k := 0 to n do begin                                               4
    t_k ← ( L[k−1]+1, k, −1 ); /* −1 は未設定を示す */                    5
    b̄_k ← ( L[k]+1, −1, k ); /* −1 は未設定を示す */                     6
    if ( |r_k| ≤ L[k] ) begin                                            7
      b̄_k ← ϵ; /* b̄_i は存在しない */                                   8
      j ← j − 1;                                                         9
    end if;                                                              10
  end for;                                                               11
  /* Step 3–2 : 𝒢_T の構築 */                                            12
  stack.clean();                                                         13
  for i := 0 to n do begin                                               14
    stack.push(i);                                                       15
    while ( stack.is_empty() ≠ true ) begin                              16
      k ← stack.pop();                                                   17
      if ( |t_k| > L[i] )                                                18
        𝒞(t_k).max ← i;                                                  19
      else begin                                                         20
        stack.push(k);                                                   21
        break;                                                           22
      end if;                                                            23
    end while;                                                           24
  end for;                                                               25
  /* Step 3–3 : 𝒢_B の構築 */                                            26
  stack.clean();                                                         27
  for i := n to 0 do begin                                               28
    if ( b̄_i ≠ ϵ )                                                      29
      stack.push(i)                                                      30
    while ( stack.is_empty() ≠ true ) begin                              31
      k ← stack.pop();                                                   32
      if ( |b̄_k| > L[i−1] ) begin                                        33
        𝒞(b̄_k).min ← i;                                                  34
```

```
      b_j ← b̄_k;                                          35
      j ← j − 1;                                          36
    else begin                                            37
      stack.push(k);                                      38
      break;                                              39
    end if;                                               40
  end while;                                              41
end for;                                                  42
/* Step 3–4 : G の構築 */                                 43
G ← merge_sort(G_T, G_B);                                 44
return G ;                                                45
end.                                                      46
```

辞書式順序昇順で整列するためには，系列を比較する必要がある．アルゴリズムでは直接扱っていないが，系列の比較は$\mathcal{C}(\cdot).\min$と$\mathcal{C}(\cdot).\max$を用いている．この2つの値で系列を比較できることを，次の命題 2 と命題 3 で示す.

命題 2 任意の$\boldsymbol{u}, \boldsymbol{v} \in \mathcal{D}(\boldsymbol{x})$に対して，$\mathcal{C}(\boldsymbol{u}).\min < \mathcal{C}(\boldsymbol{v}).\min$ならば，

$$\boldsymbol{u} \prec \boldsymbol{v}.$$

証明 最初に，$i = \mathcal{C}(\boldsymbol{u}).\min, j = \mathcal{C}(\boldsymbol{v}).\min$とおく．(2.6) から，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i), \boldsymbol{v} \in \mathcal{P}(\boldsymbol{r}_j)$である．さらに，(2.6) と$i < j$から，$i \notin \mathcal{C}(\boldsymbol{v})$. したがって，$\boldsymbol{u} \notin \mathcal{P}(\boldsymbol{r}_j)$となる．

すなわち，$\boldsymbol{r}_i \prec \boldsymbol{r}_j$なので，$\boldsymbol{u} \prec \boldsymbol{v}$である．

□

命題 3 任意の$\boldsymbol{u}, \boldsymbol{v} \in \mathcal{D}(\boldsymbol{x})$に対して，$\mathcal{C}(\boldsymbol{u}).\min = \mathcal{C}(\boldsymbol{v}).\min, \mathcal{C}(\boldsymbol{u}).\max > \mathcal{C}(\boldsymbol{v}).\max$ならば，

$$\boldsymbol{u} \prec \boldsymbol{v}.$$

証明 最初に，(2.9) から，$\mathcal{C}(\boldsymbol{v}) \subsetneq \mathcal{C}(\boldsymbol{u})$である．ここで，背理法を使って証明する．いま，$|\boldsymbol{u}| \geq |\boldsymbol{v}|$と仮定する．このとき，ある整数$i$が存在し，$i \in \mathcal{C}(\boldsymbol{u}) \cap \mathcal{C}(\boldsymbol{v})$とおける．すなわち，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i), \boldsymbol{v} \in \mathcal{P}(\boldsymbol{r}_i)$である．よって，$|\boldsymbol{u}| \geq |\boldsymbol{v}|$から，$\mathcal{C}(\boldsymbol{v}) \subset \mathcal{C}(\boldsymbol{u})$となる．しかしながら，これは，$\mathcal{C}(\boldsymbol{v}) \subsetneq \mathcal{C}(\boldsymbol{u})$に矛盾する．したがって，$|\boldsymbol{u}| < |\boldsymbol{v}|$である．

そして，$|\boldsymbol{u}| < |\boldsymbol{v}|$ と $\boldsymbol{u} \in \mathcal{P}(\boldsymbol{r}_i), \boldsymbol{v} \in \mathcal{P}(\boldsymbol{r}_i)$ から，$\boldsymbol{u} \in \mathcal{P}(\boldsymbol{v})$ となる．
以上より，$\boldsymbol{u} \prec \boldsymbol{v}$ である．

□

集合 $\mathcal{G}_{\mathrm{T}}$ の構築は，Step 3–1 と 3–2 で行われる．このステップにて，$\bar{\boldsymbol{t}}_k$ を用いずに $\boldsymbol{t}_k$ を直接求めているのは，系 2 と命題 2 から，$\bar{\boldsymbol{t}}_0 \prec \bar{\boldsymbol{t}}_1 \prec \ldots \prec \bar{\boldsymbol{t}}_n$ だからである．すなわち，$\boldsymbol{t}_k = \bar{\boldsymbol{t}}_k$ である．(4.12) の 3 要素のうち，Step 3–1 では，(4.7) から $|\boldsymbol{t}_k|$ を，系 2 から $\mathcal{C}(\boldsymbol{t}_k)$.min を求めている．

そして，Step 3–2 で，$\mathcal{C}(\boldsymbol{t}_k)$.max を求めている．$\mathcal{C}(\boldsymbol{t}_k)$.max は系 2 で与えられる．すなわち，$\mathcal{C}(\boldsymbol{t}_k)$.max を求めるには，$k$ 以上で最初に L の値が $|\boldsymbol{t}_k|$ より小さくなるインデックスを探せばよい．これは，直近上位要素発見問題 (Nearest Larger Neighbor Problem)[22] である．

そこで，Finding Nearest Larger Neighborhood Algorithm[22] を Step 3–2 の 13 行目から 25 行目で用いている．このアルゴリズムではスタックを活用し，i を 0 から n まで増やしながら $|\boldsymbol{t}_k| = i$ となる k の値を探している．これにより，1 回の走査で全ての $\boldsymbol{t}_k$ を求めることができる．

集合 $\mathcal{G}_{\mathrm{B}}$ の構築は，Step 3–1 と 3–3 で行われる．Step 3–1 では，(4.8) から $|\boldsymbol{b}_k|$ を，系 3 から $\mathcal{C}(\boldsymbol{b}_k)$.min を求めている．そして，Step 3–3 で，$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.min を求めている．$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.min は系 3 で与えられる．これも，$k$ 以下で最初に L の値が $|\boldsymbol{b}_k|$ より小さくなるインデックスを探す直近上位要素発見問題である．そこで，i を n から 0 まで減らしながら L を走査する Finding Nearest Larger Neighborhood Algorithm を Step 3–3 の 27 行目から 42 行目で用いている．

ところで，i を n から 0 に向かって走査するため，$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.min の値が大きい順に求めることになる．しかも，$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.min が同じ値の場合は，$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.max の値が小さい順に求める．すなわち，命題 2 と命題 3 から，$\mathcal{C}\left(\bar{\boldsymbol{b}}_k\right)$.min の値を求める順番は，$\bar{\boldsymbol{b}}_k$ の辞書式順序降順である．この性質を利用するために，Finding Nearest Larger Neighborhood Algorithm に 35, 36 行目の処理を加え，辞書式順序昇順に整列した $\boldsymbol{b}_k$ を求めている．

4.7 Step 4のアルゴリズム

Step 4 では，Step 3 で求めた集合 $\mathcal{G}$ の全ての要素 $\boldsymbol{g}_i$ $(0 \leq i \leq |\mathcal{G}| - 1)$ に対して，(4.1) で定義した $\mathcal{H}(\boldsymbol{g}_i)$ を求めて，反辞書を構築する．(4.1) より，$\boldsymbol{g}_i \neq \lambda$ の場合に $\mathcal{H}(\boldsymbol{g}_i)$ を求めるには，(2.10) で定義した全接頭記号集合 $\mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{g}_i))$ と

$\mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)$ が必要である．ここで，

$$l\,(i) = \max\{j | j < i, |\boldsymbol{g}_j| < |\boldsymbol{g}_i|\} \tag{4.18}$$

を定義し，次の補題 3 と命題 4 を示す.

補題 3 任意の $\boldsymbol{g}_i\ (\neq \lambda)$ に対して，

$$\boldsymbol{g}_{l(i)} \in \mathcal{P}(\boldsymbol{g}_i)\,.$$

証明 背理法を使って証明する．いま，$\boldsymbol{g}_{l(i)} \notin \mathcal{P}(\boldsymbol{g}_i)$ と仮定する．このとき，ある記号 $\alpha, \beta \in \mathcal{X}(\alpha < \beta)$ と系列 $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \mathcal{X}^*(|\boldsymbol{u}| < |\boldsymbol{v}|)$ が存在し，$\boldsymbol{g}_{l(i)} = \boldsymbol{w}\alpha\boldsymbol{u}, \boldsymbol{g}_i = \boldsymbol{w}\beta\boldsymbol{v}$ とおける．よって，(4.6) と補題 2 から，ある整数 j が存在し，$\boldsymbol{g}_j = \boldsymbol{w}\beta$. このとき，$|\boldsymbol{g}_j| < |\boldsymbol{g}_i|, \boldsymbol{g}_{l(i)} \prec \boldsymbol{g}_j \prec \boldsymbol{g}_i$ から，$l\,(i) < j < i$ である．しかしながら，これは，(4.18) に矛盾する．よって，$\boldsymbol{g}_{l(i)} \in \mathcal{P}(\boldsymbol{g}_i)$ である．

□

命題 4 任意の $\boldsymbol{g}_i\ (\neq \lambda)$ に対して，

$$\mathcal{X}^{\mathrm{H}}(\pi(\boldsymbol{g}_i)) = \mathcal{X}^{\mathrm{H}}\Big(\boldsymbol{g}_{l(i)}\Big)\,.$$

証明 表記を簡潔にするために，$\boldsymbol{u} = \boldsymbol{g}_{l(i)}, \boldsymbol{v} = \pi(\boldsymbol{g}_i)$ とおく．(4.18) から，$|\boldsymbol{u}| \leq |\boldsymbol{v}|$ である．そこで，$|\boldsymbol{u}| = |\boldsymbol{v}|$ と $|\boldsymbol{u}| < |\boldsymbol{v}|$ の 2 つの場合に分けて証明する．

最初に，$|\boldsymbol{u}| = |\boldsymbol{v}|$ の場合を証明する．補題 3 と $\boldsymbol{v} \in \mathcal{P}(\boldsymbol{g}_i)$ から，$\boldsymbol{u} = \boldsymbol{v}$. したがって，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \mathcal{X}^{\mathrm{H}}(\boldsymbol{v})$ である．

次に，$|\boldsymbol{u}| < |\boldsymbol{v}|$ の場合を証明する．

補題 3 と (4.18) から，$\boldsymbol{u}$ は $\mathcal{P}(\boldsymbol{g}_i) \cap \{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-1}\}$ の最長系列である．

ここで，背理法を使って証明する．いま，$\mathcal{C}(\boldsymbol{u}) \setminus \mathcal{C}(\boldsymbol{v}) \neq \phi$ と仮定する．このとき，ある整数 j, k が存在し，$j \in \mathcal{C}(\boldsymbol{u}) \cap \mathcal{C}(\boldsymbol{v})\,, k \in \mathcal{C}(\boldsymbol{u}) \setminus \mathcal{C}(\boldsymbol{v})$ とおける．すなわち，(2.6) と補題 3 から，$|\boldsymbol{u}| \leq \mathrm{lcp}\,(\boldsymbol{r}_j, \boldsymbol{r}_k) \leq |\boldsymbol{v}| - 1$ となる．したがって，(4.6) から，ある $\boldsymbol{w} \in \mathcal{X}^*$ が存在し，$\boldsymbol{w} \in \mathcal{E}(j)\,, |\boldsymbol{w}| = \mathrm{lcp}\,(\boldsymbol{r}_j, \boldsymbol{r}_k) + 1$ とおける．このとき，補題 2 から，$\boldsymbol{w} \in \mathcal{G}$ である．ところで，$|\boldsymbol{u}| < |\boldsymbol{w}| < |\boldsymbol{g}_i|$ と $\boldsymbol{u}, \boldsymbol{w} \in \mathcal{P}(\boldsymbol{g}_i)$ から，$\boldsymbol{u} \prec \boldsymbol{w} \prec \boldsymbol{g}_i$ である．よって，$\boldsymbol{w} \in \{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-1}\}$. しかしながら，これは，$\boldsymbol{u}$ が $\mathcal{P}(\boldsymbol{g}_i) \cap \{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-1}\}$ の最長系列であることに矛盾する．したがって，$\mathcal{C}(\boldsymbol{u}) \setminus \mathcal{C}(\boldsymbol{v}) = \phi$ である．すなわち，(2.11) から，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{u}) = \mathcal{X}^{\mathrm{H}}(\boldsymbol{v})$ である．

以上より，題意が成り立つ．

□

系 4 任意の $\boldsymbol{g}_i$ $(\neq \lambda)$ に対して，

$$\mathcal{H}(\boldsymbol{g}_i) = \mathcal{X}^{\mathrm{H}}\Big(\boldsymbol{g}_{l(i)}\Big) \setminus \mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)\,.$$

Step 4 のアルゴリズム construct_antidictionary を以下に示す．アルゴリズム中の $\Lambda[\cdot]$ は，$\boldsymbol{x}$ のインデックスを格納する長さ $|\mathcal{X}|$ の配列である．stack は，$\boldsymbol{g}_i$ の添え字を格納するスタックである．

なお，$\mathcal{X}^{\mathrm{H}}(i)$ に記号 α を追加する処理を $\mathcal{X}^{\mathrm{H}}(i)$ .add(α) と記す．同様にして，$\mathcal{H}(\boldsymbol{u})$ に記号 α を追加する処理を $\mathcal{H}(\boldsymbol{u})$ .add(α) と記す．$\mathcal{A}(\boldsymbol{x})$ に系列 $\boldsymbol{u}$ を追加する処理を $\mathcal{A}(\boldsymbol{x})$ .add$(\boldsymbol{u})$ と記す．

Algorithm construct_antidictionary

input : 入力系列 $\boldsymbol{x}$, 接尾辞配列 S, 系列の集合 $\mathcal{G}$

output : 反辞書 $\mathcal{A}(\boldsymbol{x})$

```
begin
  /*Step 4-1 : 初期設定*/
  for ( ∀α ∈ 𝒳 ) do
    Λ[α] ← n + 1;
  /*Step 4-2 : 𝒳^H(g_i) の構築*/
  i ← |𝒢| − 1
  for k := n to 0 do begin
    if ( S[k] − 1 ≥ 1 )
      Λ[x_{S[k]−1}] ← k; /*最終インデックスの更新*/
    while ( i ≥ 0 and 𝒞(g_i).min = k ) begin
      for ( ∀α ∈ 𝒳 ) do begin
        if ( Λ[α] ≤ 𝒞(g_i).max )
          𝒳^H(g_i) .add(α);
      end for;
      i ← i − 1;
    end while;
  end for;
  /*Step 4-3 : ℋ(g_i) の構築*/
  /*Step 4-3.1 : ℋ(g_0) の設定*/
  for ( ∀α ∈ 𝒳 ) do begin
    if (Λ[α] = n + 1)
      ℋ(g_0) .add(α); /*ℋ(λ)*/
```

end for; 23

/∗ *Step 4–3.2 : $l\,(i)$* **の設定** ∗/ 24

$p \leftarrow 0$ 25

stack.clean(); 26

for $i := 1$ **to** $|\mathcal{G}| - 1$ **do begin** 27

while (stack.is_empty()$\neq$**true and** $|\boldsymbol{g}_p| \geq |\boldsymbol{g}_i|$) 28

$p \leftarrow$ stack.pop(); 29

$l\,(i) \leftarrow p$; 30

stack.push(p); 31

$p \leftarrow i$; 32

end for; 33

/∗ *Step 4–3.3 : $\mathcal{H}(\boldsymbol{g}_i)$ $(i \geq 1)$* **の設定** ∗/ 34

for $i := 1$ **to** $|\mathcal{G}| - 1$ **do** 35

$\mathcal{H}(\boldsymbol{g}_i) \leftarrow \mathcal{X}^{\mathrm{H}}\Big(\boldsymbol{g}_{l(i)}\Big) \setminus \mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)$; 36

/∗ *Step 4–4 : $\mathcal{A}(\boldsymbol{x})$* **の構築** ∗/ 37

for $i := 0$ **to** $|\mathcal{G}| - 1$ **do** 38

for ($\forall \alpha \in \mathcal{H}(\boldsymbol{g}_i)$) **do** 39

$\mathcal{A}(\boldsymbol{x})$ .add($\alpha \boldsymbol{g}_i$); 40

return $\mathcal{A}(\boldsymbol{x})$; 41

end. 42

Step 4–2 では，k を n から 0 まで減らしながら，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)$ $(0 \leq i \leq |\mathcal{G}| - 1)$ を構築している．k が減るたびに記号 $x_{\mathsf{S}[k]-1}$ を調べ，$\mathsf{\Lambda}\left[x_{\mathsf{S}[k]-1}\right]$ に k を格納する．すなわち，k の値を固定したとき，

$$\mathsf{\Lambda}\,[\alpha] = \min\Big\{j | x_{\mathsf{S}[j]-1} = \alpha, k \leq j \leq n\Big\} \ (\alpha \in \mathcal{X})$$

である．そして，(2.11) から，$\mathcal{C}(\boldsymbol{g}_i).\min = k$ となる全ての $\boldsymbol{g}_i$ について，それぞれ $k \leq \mathsf{\Lambda}\,[\alpha] \leq \mathcal{C}(\boldsymbol{g}_i).\max$ を満たす α を調べて，$\mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)$ を求める．

Step 4–3.1 では，$\mathcal{H}(\boldsymbol{g}_0)$，すなわち $\mathcal{H}(\lambda)$ を求めている．これは，(4.1) から，未出現記号の集合である．そこで，$\mathsf{\Lambda}\,[\alpha]$ を調べて求める．また，Step 4–3.2 では (4.18) の $l\,(i)$ を求め，Step 4–3.3 で $\mathcal{H}(\boldsymbol{g}_i)$ を求める．

最後に Step 4–4 で，定理 1 から，$\mathcal{H}(\boldsymbol{g}_i)$ と $\boldsymbol{g}_i$ を用いて全ての極小禁止語を求め，反辞書 $\mathcal{A}(\boldsymbol{x})$ を構築する．

4.8 計算時間の評価

提案手法SA2ADは，Step 1からStep 4までの4ステップで構成されている．このうちStep 1とStep 2は，計算時間と記憶量が線形の既知の手法[16] – [20], [11], [12]を用いる．Step 1とStep 2の計算時間をそれぞれ$T_1(n), T_2(n)$とおくと，$T_1(n) = O(n), T_2(n) = O(n)$である．

ここで，非負整数上に定義された実関数$f(n), g(n)$に対して，$f(n) = O(g(n))$であるとは，ある定数$E > 0$とある$n_0 \in \mathbb{N}$が任意の$n > n_0$に対して，

$$|f(n)| \leq E|g(n)|$$

が成立することとする．

Step 3の計算時間を$T_3(n)$とおくと，次の補題4を示せる．

補題 4 入力系列$\boldsymbol{x}$の長さnに対して，

$$T_3(n) = O(n).$$

証明 Step 3のアルゴリズムconstruct_GのStep 3–1, 3–2, 3–3, 3–4の計算時間をそれぞれ$T_{31}, T_{32}, T_{33}, T_{34}$とおく．このうち，$T_{31} = O(n)$は自明である．

Step 3–2の16–24行目にあるwhileループは，stackが空でないときに繰り返す．また，ループ中にstack.pop()を必ず行う．したがって，Step 3–2の処理が終わるまでにwhileループを繰り返す総回数の上限は，Step 3–2の処理が終わるまでにstack.push(·)した総回数に等しい．Step 3–2は，全体がforループとなっており，1回のループにつき最大で2回stack.push(·)する．したがって，whileループを繰り返す総回数の上限は$2n + 2$. よって，$T_{32} = O(n)$.

Step 3–3も同様に，31–41行目のwhileループを繰り返す総回数の上限は$2n + 2$. よって，$T_{33} = O(n)$.

Step 3–4では，$\mathcal{G}_{\mathrm{T}}$と$\mathcal{G}_{\mathrm{B}}$のマージソートを行っている．このステップまでに$\mathcal{G}_{\mathrm{T}}, \mathcal{G}_{\mathrm{B}}$はソート済みである．(4.13), (4.14)から，$|\mathcal{G}_{\mathrm{T}}| \leq n + 1, |\mathcal{G}_{\mathrm{B}}| \leq n + 1$. よって，$T_{34} = O(n)$.

以上より，$T_3(n) = T_{31} + T_{32} + T_{33} + T_{34} = O(n)$である．

□

Step 4の計算時間を$T_4(n)$とおくと，次の補題5を示せる．

補題 5 入力系列$\boldsymbol{x}$の長さnに対して，

$$T_4(n) = O(n).$$

証明 Step 4 のアルゴリズム construct_antidictionary の Step 4–1, 4–2, 4–3, 4–4 の計算時間をそれぞれ $T_{41}, T_{42}, T_{43}, T_{44}$ とおく．このうち，$T_{41} = O(1)$ は自明である．(4.9), (4.10) から，$|\mathcal{G}| \leq 2n + 2$. よって，$T_{44} = O(n)$.

Step 4–2 の 10–16 行目にある while ループは，i が負のときは繰り返さない．また，i の初期値は $|\mathcal{G}| - 1$ で，ループ中に必ず i の値を -1 する．$|\mathcal{G}| \leq 2n+2$ から，Step 4–2 の処理が終わるまでに while ループを繰り返す総回数の上限は，$2n + 2$. よって，$T_{42} = O(n)$.

次に，Step 4–3.1, 4–3.2, 4–3.3 の計算時間をそれぞれ $T_{43.1}, T_{43.2}, T_{43.3}$ とおく．このうち，$T_{43.1} = O(1)$ は自明である．さらに，$|\mathcal{G}| \leq 2n+2$ から，$T_{43.3} = O(n)$. Step 4–3.2 の 28–29 行目にある while ループは，stack が空のときは繰り返さない．また，ループ中に必ず stack.pop() する．したがって，Step 4–3.2 の処理が終わるまでに while ループを繰り返す回数の上限は，Step 4–3.2 の処理が終わるまでに stack.push(·) した総回数に等しい．Step 3–2 は，全体が for ループとなっており，1 回のループにつき 1 回 stack.push(·) する．したがって，while ループを繰り返す総回数の上限は $n + 1$. よって，$T_{43.2} = O(n)$. すなわち，$T_{43} = T_{43.1} + T_{43.2} + T_{43.3}$ から，$T_{43} = O(n)$.

以上より，$T_4(n) = T_{41} + T_{42} + T_{43} + T_{44} = O(n)$ である．

□

提案アルゴリズム SA2AD の計算時間を $T(n)$ とおくと，次の定理 3 を示せる．

定理 3 入力系列 $\boldsymbol{x}$ の長さ n に対して，

$$T(n) = O(n).$$

証明 補題 4，補題 5 と，$T(n) = T_1(n) + T_2(n) + T_3(n) + T_4(n)$ から，$T(n) = O(n)$ である．

□

4.9 記憶量の評価

アルゴリズム SA2AD の Step 1, Step 2, Step 3, Step 4 のそれぞれの記憶量を $M_1(n), M_2(n), M_3(n), M_4(n)$ とおく．なお，記号とインデックス，そして系列長は固定幅の記憶量で格納するので，記憶量を定数 c とする．なお，系列は (4.12) で表現しているので，記憶量を定数 $3c$ とする．

Step 1 と Step 2 は，計算時間と記憶量が $O(n)$ の既知の手法 [16] – [20], [11], [12] を用いるから，$M_1(n) = O(n), M_2(n) = O(n)$ である．

Step 3 のアルゴリズム construct_G の記憶量を，次の補題 6 で示す.

補題 6 入力系列 $\boldsymbol{x}$ の長さ n に対して，

$$M_3(n) = O(n).$$

証明 全ての $\boldsymbol{t}_i \in \mathcal{G}_{\mathrm{T}}$ を格納する記憶量は，(4.15) から，$3(n+1)c$. 全ての $\boldsymbol{b}_i, \bar{\boldsymbol{b}}_i \in \mathcal{G}_{\mathrm{B}}$ を格納する記憶量は，(4.14), (4.16) から，高々$6(n+1)c$. そして，stack には，接尾辞 S のインデックスを重複することなく格納するから，記憶量は高々$(n+1)c$. また，(4.10) から，$|\mathcal{G}| \leq 2(n+1)$. よって，$\mathcal{G}$ を格納する記憶量は高々$6(n+1)c$.

以上より，$M_3(n) \leq 16(n+1)c$. よって，$M_3(n) = O(n)$ である.

□

Step 4 のアルゴリズム construct_antidictionary の記憶量を，次の補題 7 で示す.

補題 7 入力系列 $\boldsymbol{x}$ の長さ n に対して，

$$M_4(n) = O(n).$$

証明 配列 Λ は，長さ $|\mathcal{X}|$ でインデックスを格納するから，記憶量は $|\mathcal{X}|c$. そして，$\mathcal{G}$ を格納する記憶量は，(4.10) から，高々$6(n+1)c$.

また，$\mathcal{X}^{\mathrm{H}}(\cdot)$ は高々$|\mathcal{X}|$ 個の記号からなる集合である．すなわち，$\boldsymbol{g}_i \in \mathcal{G}$ ごとに $\mathcal{X}^{\mathrm{H}}(\boldsymbol{g}_i)$ を格納するので，記憶量は高々$6|\mathcal{X}|(n+1)c$. なお，$\mathcal{H}(\cdot)$ も高々$|\mathcal{X}|$ 個の記号からなる集合である．そして，$\boldsymbol{g}_i$ ごとに格納するから，記憶量は高々$6|\mathcal{X}|(n+1)c$.

つぎに，$l(i)$ は，長さ $|\mathcal{G}|$ の配列でインデックスを格納するから，記憶量は高々$2(n+1)c$. さらに，stack は，インデックスを格納し，最大で $|\mathcal{G}|$ 回の stack.push$(\cdot)$ を実行するから，記憶量は高々$2(n+1)c$. また，$\mathcal{A}(\boldsymbol{x})$ は，記号と系列を格納し，最大で $|\mathcal{X}||\mathcal{G}|$ 個の要素を加えるから，記憶量は高々$8|\mathcal{X}|(n+1)c$.

以上より，$M_4(n) \leq (20|\mathcal{X}|+10)(n+1)c+|\mathcal{X}|c$. よって，$M_4(n) = O(n)$ である.

□

提案アルゴリズム SA2AD の記憶量を $M(n)$ とおくと，次の定理 4 を示せる.

定理 4 入力系列 $\boldsymbol{x}$ の長さ n に対して，

$$M(n) = O(n).$$

証明 補題 6，補題 7 と，$M(n) = M_1(n)+M_2(n)+M_3(n)+M_4(n)$ から，$M(n) = O(n)$ である.

□

4.10 計算機実験

提案手法と接尾辞木を用いた手法とを，計算機実験で比較した．計算機のCPUはCore2 Duo E8500 (3.16GHz, 2コア)，メモリ量は4GByte，OSはWindows XP SP3 32bit版である．プログラムは両方ともC++言語で作成した．

入力系列は，$|\mathcal{X}| = 256$の独立同一分布をもつ情報源で，各記号の生起確率は等確率である．入力系列長は，50×10^3ごとに用意した．それぞれ計算時間と記憶量を3回測定し，平均を求めた．なお，OSの制限でアプリケーションが使用できる最大メモリ量は2Gbyteとなっている．そのため，接尾辞木を用いた手法では入力系列長200×10^3で，提案手法では入力系列長450×10^3で，使用メモリ量が多すぎて実行不可能となった．

計算時間の比較結果を図4.1に，記憶量の比較結果を図4.2に示す．

提案手法の計算時間と記憶量の両方とも，$O(n)$であることが計算機実験でも示された．また，接尾辞木を用いた手法と比べて，提案手法の計算量は，計算時間が約20分の1に，記憶量が約2.5分の1に改善された．

さらに，記憶量の内訳について調べた．

接尾辞木を用いた手法では，記憶量の約99%がMF–linkの管理に使われていた．MF–linkとは，接尾辞木の中間ノードのうち，枝を2本以上持つ中間ノードから伸びる特別なリンクである．接尾辞木にMF–linkを追加することで反辞書を構築する．MF–linkを追加した中間ノードからは，高々$|\mathcal{X}|$本のMF–linkが伸びているため，MF–linkの管理に使用する記憶量は$|\mathcal{X}|n$に比例する．今回の計算機実験において，MF–linkの管理に使用している記憶量は約$40|\mathcal{X}|n = 10000n$ byteであった．

一方，提案手法では，記憶量の約94〜95%が$\mathcal{A}(\boldsymbol{x})$の記憶に使われていた．$\mathcal{A}(\boldsymbol{x})$に格納する極小禁止語は，先頭記号と後続する系列に分けて表現している．すなわち，$\mathcal{A}(\boldsymbol{x})$に使用する記憶量も$|\mathcal{X}|n$に比例する．今回の計算機実験において，$\mathcal{A}(\boldsymbol{x})$に使用している記憶量は，約$14.8|\mathcal{X}|n \sim 16.4|\mathcal{X}|n = 3800n \sim 4200n$ byteとなった．すなわち，提案方式で改善された記憶量は，MF–linkの管理に使用している記憶量と$\mathcal{A}(\boldsymbol{x})$の記憶量との差であることがわかった．

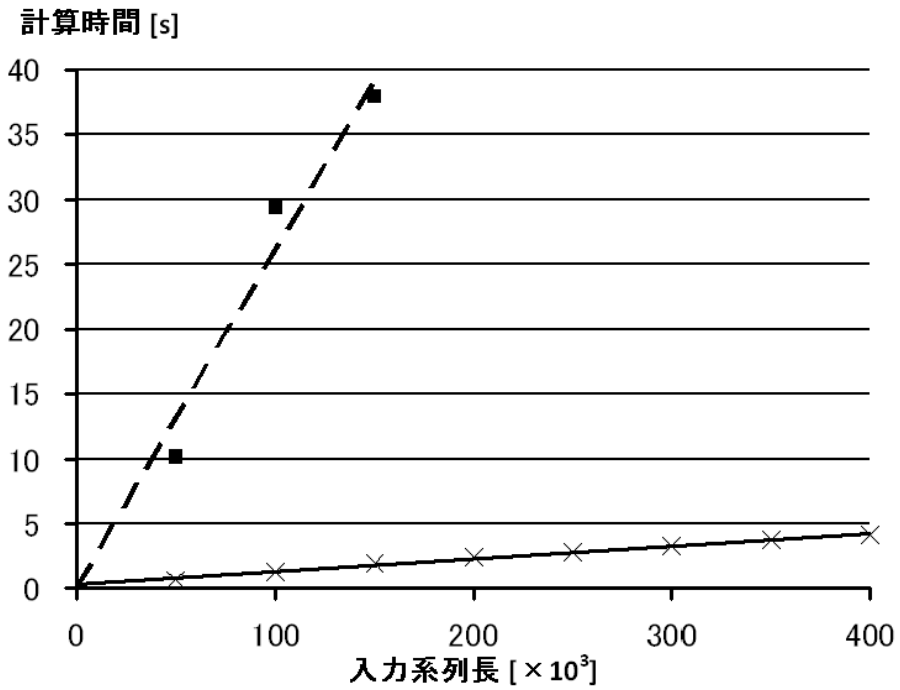


図 4.1: 計算時間の比較（実線：提案手法，点線：接尾辞木を用いた手法）

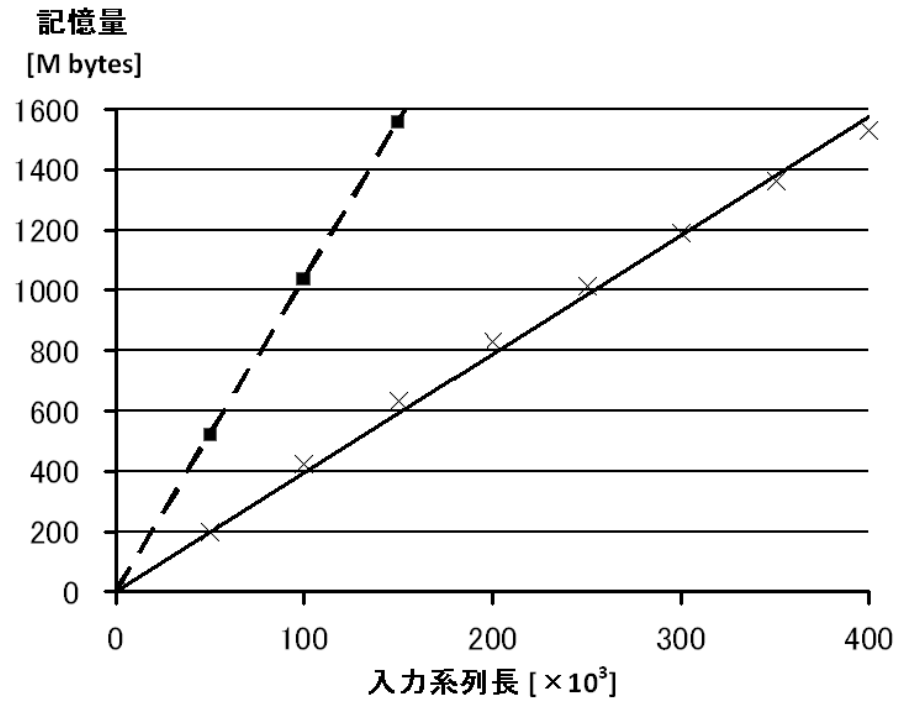


図 4.2: 記憶量の比較（実線：提案手法，点線：接尾辞木を用いた手法）

次に，アルファベットサイズ $m\ (= |\mathcal{X}|)$ を変化させたときの計算時間と記憶量の変化を測定した．用意したのは，m がそれぞれ $2, 50, 100, 150, 200, 250$ の独立同一分布をもつ情報源で，各記号の生起確率は等確率である．入力系列長は，50×10^3 ごとに，系列長 50×10^3〜250×10^3 まで用意した．それぞれ計算時間と記憶量を 100 回測定し，平均を求めた．

計算時間の測定結果を図 4.3 に，記憶量の測定結果を図 4.4 に示す．

いずれのアルファベットサイズでも，計算時間と記憶量は入力系列長に対して線形であることが実験結果でも示せた．また，計算時間と記憶量はアルファベットサイズにも比例することがわかる．これは，Step 4 のアルゴリズム construct_antidictionary の 3〜4 行目，11〜14 行目，20〜23 行目に m 回繰り返す処理があるためである．さらに，36 行目の $\mathcal{H}(\boldsymbol{g}_i)$ を求める処理と 39 行目の **for** 文の実装においても，最大で m 回繰り返す処理を用いている．

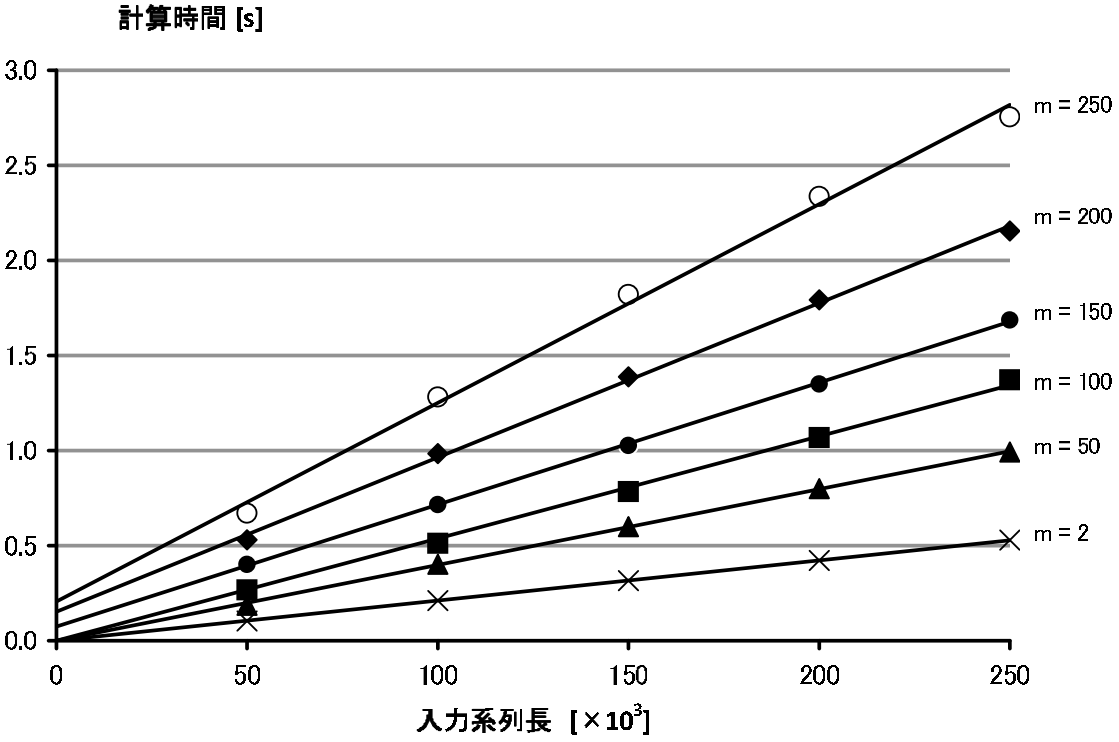


図 4.3: アルファベットサイズ m ごとの計算時間

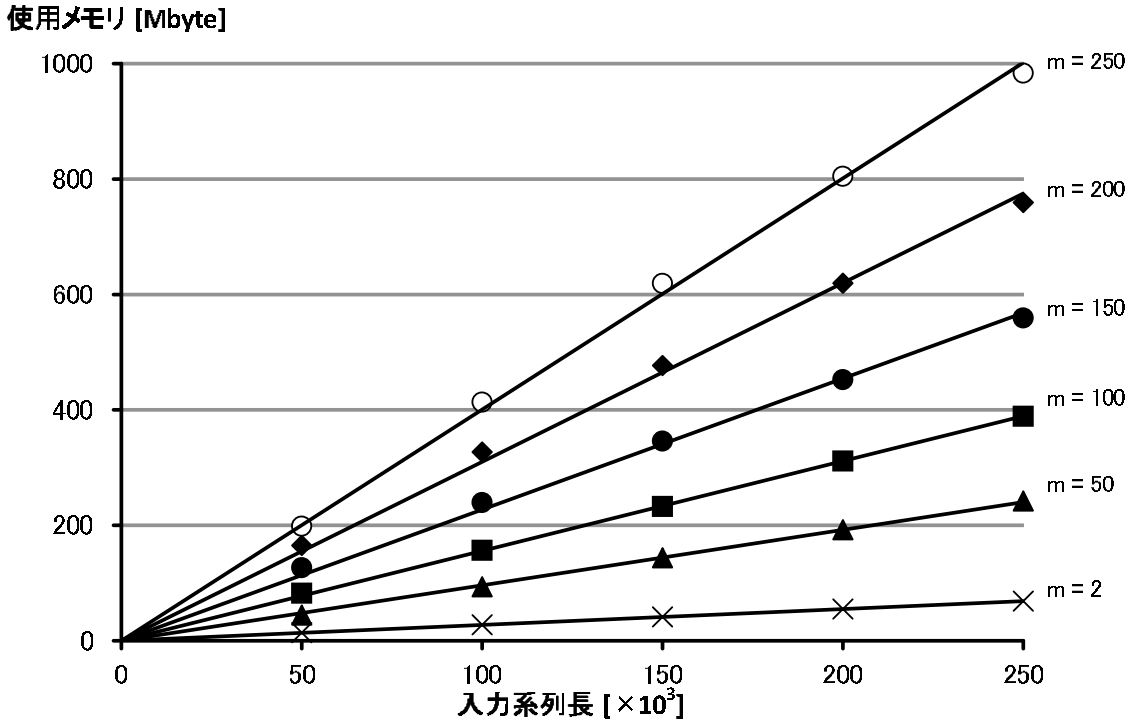


図 4.4: アルファベットサイズ m ごとの記憶量

第5章　動的反辞書構築の提案

5.1　静的反辞書構築と動的反辞書構築

静的反辞書構築とは，入力系列を一度すべて読み込み終わってから反辞書を構築することである．第4章で提案した手法も静的反辞書構築である．接尾辞配列やL配列のように，既存のデータ構造の多くは入力系列をすべて読み込み終わった状態から構築するアルゴリズムが多い．そのため，反辞書を静的に構築する場合には，これらのデータ構造を活用できる利点がある．

ただし，反辞書を使って入力系列自身を圧縮したり，調べたりするときには，静的反辞書構築ではオーバーヘッドを生じることになる．例えば，データ圧縮の反辞書法[5]では，圧縮系列に加えて，データ圧縮で使う反辞書オートマトンをオーバーヘッドとして復号側へ送る．不整脈検出[8]では，学習系列を準備して反辞書オートマトンを作成する．

一方，動的反辞書構築とは，入力系列を1記号ずつ読み込みながら逐次的に反辞書を構築することである．入力記号に合わせながら反辞書を変化させていくので，上記のオーバーヘッドを無くすことができる．また，1記号読み込んだ後の反辞書の変化が少ない場合，第4章の提案手法よりも効率がよくなる可能性がある．

この章では，記号の入力ごとに，逐次的に反辞書オートマトンを変化させていく新しい手法を提案する．反辞書オートマトンについては，5.3節に後述する．

5.2　表記の拡張

入力系列

$$\boldsymbol{x}_1^n = x_1 x_2 \cdots x_n$$

は，記号x_{n+1}の入力によって系列の末尾に記号x_{n+1}が加わった系列

$$\boldsymbol{x}_1^{n+1} = x_1 x_2 \cdots x_n x_{n+1}$$

に変化する．本章までは静的反辞書構築を扱っていたので，便宜上，n を意識しない形で表記していた．しかし，この章からは記号 x_{n+1} を入力する動的反辞書構築なので，以下のように n を明示する形に表記を拡張する．

- 系列 $\boldsymbol{x}_1^n$ を，$\boldsymbol{x}^n$ と表記する．
- 第 2 章の (2.10) で定義した，系列 $\boldsymbol{x}^n$ 上の $\boldsymbol{u}$ の全接頭記号集合 $\mathcal{X}^{\mathrm{H}}(\boldsymbol{u})$ を，$\mathcal{X}_n^{\mathrm{H}}(\boldsymbol{u})$ と表記する．
- 第 2 章の (2.12) で定義した，系列 $\boldsymbol{x}^n$ 上の $\boldsymbol{u}$ の全接尾記号集合 $\mathcal{X}^{\mathrm{T}}(\boldsymbol{u})$ を，$\mathcal{X}_n^{\mathrm{T}}(\boldsymbol{u})$ と表記する．
- 第 4 章の (4.1) で定義した，系列 $\boldsymbol{x}^n$ 上の $\mathcal{H}(\boldsymbol{u})$ を，$\mathcal{H}_n(\boldsymbol{u})$ と表記する．
- 第 4 章の (4.3) で定義した，系列 $\boldsymbol{x}^n$ 上の $\mathcal{T}(\boldsymbol{u})$ を，$\mathcal{T}_n(\boldsymbol{u})$ と表記する．

5.3 反辞書オートマトン

反辞書オートマトンとは，極小禁止語を受理する決定性有限オートマトンである．反辞書法 [5] で用いられている．

図 5.1 に $\boldsymbol{x}^{10} = \mathsf{abcaababac}$ から構築した反辞書オートマトンを示す．このとき，

$$\mathcal{A}\left(\boldsymbol{x}^{10}\right) = \{\mathsf{aaa}, \mathsf{aabac}, \mathsf{aabc}, \mathsf{aac}, \mathsf{aca}, \mathsf{baa}, \mathsf{babab}, \mathsf{babc}, \mathsf{bb}, \mathsf{cab}, \mathsf{cac}, \mathsf{cb}, \mathsf{cc}\}$$

である．

一番左上のノードが開始状態である．本論文では，ルートノードと呼ぶことにする．黒色のノードが極小禁止語を受理する状態である．本論文では，MFW ノードと呼ぶことにする．それ以外のノードを中間ノードと呼ぶことにする．

また，任意のノード p に対して，ルートノードから最短の遷移で到達する経路を最短パスと呼ぶことにする．さらに，最短パス上にあるラベルをルートノードから順番に接続した系列を p のラベル列と呼び，$\boldsymbol{L}(p)$ と表記する．MFW ノードのラベル列は，受理する極小禁止語を表す．

この反辞書オートマトンには，任意の系列 $\boldsymbol{v} \in \mathcal{X}^*$ を入力することができる．ルートノードから開始し，入力系列の先頭から順番に，入力記号と同じラベルを持つ遷移を進んでいく．極小禁止語を受理するか，すべての入力記号の

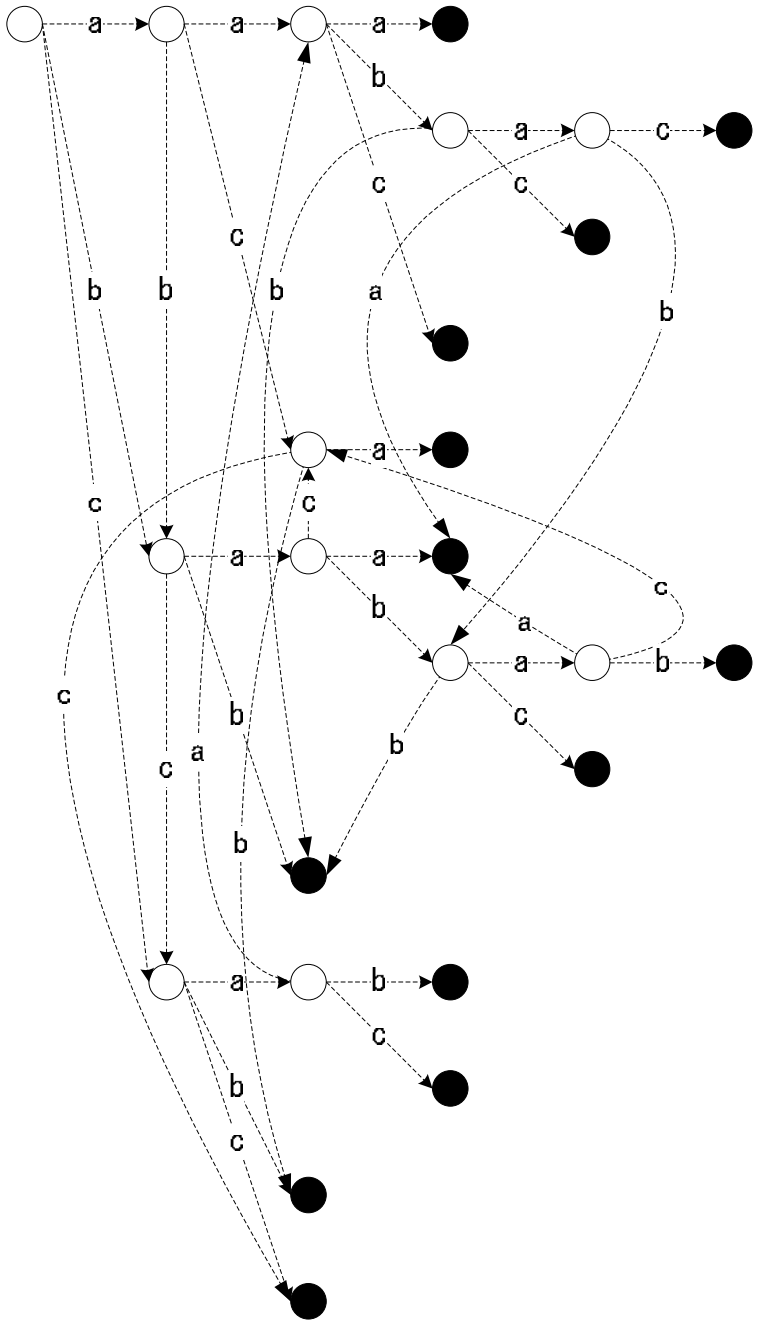


図 5.1: $\boldsymbol{x}^{10} = \mathtt{abcaababac}$ の反辞書オートマトン．黒丸のノードが極小禁止語を示す.

遷移を終了すると遷移は終了である．このとき到達したノードを$\boldsymbol{v}$のローカスと呼び，$\mathcal{N}(\boldsymbol{v})$と表記する．

反辞書オートマトンは，反辞書トライを拡張して作ることができる．すなわち，反辞書トライの葉ノード以外のノードqに，すべての記号$\beta \in \mathcal{X}$のラベルを持つ遷移を追加する．このとき，$\boldsymbol{L}(q)\beta$をラベル列に持つノードが存在しない時は$\sigma(\boldsymbol{L}(q)\beta)$をラベル列に持つノードを探して遷移する．このノードも存在しないときには，さらに$\sigma^2(\boldsymbol{L}(q)\beta)$をラベル列に持つノードを探して遷移する．以降，遷移先のノードが見つかるまでラベル列の先頭記号を削っていきながら遷移先のノードを探していく．

言いかえると，$\boldsymbol{L}(q)\beta$の接尾辞をラベル列にもつノードの中で，ラベル列の長さが最大のノードへの遷移を，反辞書トライに追加する．

なお，図5.1には書かれていないが，本論文では全てのノードpには，$\boldsymbol{L}(p)$の接尾辞をラベル列にもつノードの中で，ラベル列の長さが最大のノードへのリンクを持っている．このリンクをsuffix linkと呼ぶことにする．ノードpのsuffix link先のノードを$\mathcal{S}(p)$と表記する．

5.4 既知の結果

入力系列$\boldsymbol{x}^n$ $(n \geq 1)$の末尾に記号x_{n+1}を加えて$\boldsymbol{x}^{n+1}$になったとき，反辞書は$\mathcal{A}(\boldsymbol{x}^n)$から$\mathcal{A}(\boldsymbol{x}^{n+1})$へ変化する．このとき，$\mathcal{A}(\boldsymbol{x}^n)$から削除された極小禁止語を$\mathcal{A}_{\mathrm{DEL}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$とおき，新しく加わった極小禁止語を$\mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$とおく．すなわち，

$$\mathcal{A}_{\mathrm{DEL}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right) = \{\boldsymbol{w} | \boldsymbol{w} \in \mathcal{A}(\boldsymbol{x}^n), \boldsymbol{w} \notin \mathcal{A}\left(\boldsymbol{x}^{n+1}\right)\}, \tag{5.1}$$

$$\mathcal{A}_{\mathrm{NEW}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right) = \{\boldsymbol{w} | \boldsymbol{w} \notin \mathcal{A}(\boldsymbol{x}^n), \boldsymbol{w} \in \mathcal{A}\left(\boldsymbol{x}^{n+1}\right)\} \tag{5.2}$$

である．明らかに，$\mathcal{A}_{\mathrm{DEL}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$と$\mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$は互いに素である．また，$\mathcal{A}(\boldsymbol{x}^n)$と$\mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$も互いに素である．そして，

$$\mathcal{A}\left(\boldsymbol{x}^{n+1}\right) = \left\{\mathcal{A}(\boldsymbol{x}^n) \setminus \mathcal{A}_{\mathrm{DEL}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right)\right\} \cup \mathcal{A}_{\mathrm{NEW}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right)$$

が成り立つ．

これまでの私たちの研究成果[23], [24]では，接尾辞をタイプIとタイプIIの2種類に分割することで，$\mathcal{A}_{\mathrm{DEL}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$と$\mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$の性質を示している．

ここで，タイプIの接尾辞とは，$\boldsymbol{x}^n$ 上に1箇所だけ出現する接尾辞である．また，タイプIIの接尾辞とは，$\boldsymbol{x}^n$ 上に複数箇所出現する接尾辞である．そこで，すべてのタイプIの接尾辞の集合を，

$$\mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n) \;=\; \{\boldsymbol{u} \in \mathcal{S}(\boldsymbol{x}^n) \,|\, \boldsymbol{u} \notin \mathcal{D}\Big(\boldsymbol{x}^{n-1}\Big)\} \tag{5.3}$$

とおき，すべてのタイプIIの接尾辞の集合を，

$$\mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n) \;=\; \{\boldsymbol{u} \in \mathcal{S}(\boldsymbol{x}^n) \,|\, \boldsymbol{u} \in \mathcal{D}\Big(\boldsymbol{x}^{n-1}\Big)\} \tag{5.4}$$

とおく．なお，$\mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)$ と $\mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ は互いに素で，$\mathcal{S}(\boldsymbol{x}^n) = \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n) \cup \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ であることを注記する．さらに，$\mathcal{D}(\boldsymbol{x}^{n-1})$ と $\mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)$ は互いに素で，$\mathcal{D}(\boldsymbol{x}^n) = \mathcal{D}(\boldsymbol{x}^{n-1}) \cup \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)$ であることを注記する．また，$\boldsymbol{x}^n \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)\,, \lambda \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ である．

そして，$\boldsymbol{x}^n$ の最も短いタイプIの接尾辞を

$$\boldsymbol{s}_n = \mathrm{argmin}\,\{|\boldsymbol{v}||\boldsymbol{v} \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)\} \tag{5.5}$$

と表記し，$\boldsymbol{s}_n$ とタイプI・IIとの関係を次の命題5で示す．

命題 5 任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，接尾辞 $\boldsymbol{u} \in \mathcal{S}(\boldsymbol{x}^n)$ が，

$|\boldsymbol{u}| \geq |\boldsymbol{s}_n|$ ならば， $\boldsymbol{u} \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)\,,$

$|\boldsymbol{u}| < |\boldsymbol{s}_n|$ ならば， $\boldsymbol{u} \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)\,.$

証明 最初に，$|\boldsymbol{u}| \geq |\boldsymbol{s}_n|$ の場合を考える．このとき，$\boldsymbol{u}$ と $\boldsymbol{s}_n$ は $\boldsymbol{x}^n$ の接尾辞なので，$\boldsymbol{s}_n \in \mathcal{S}(\boldsymbol{u})$ である．よって，$\boldsymbol{s}_n \in \mathcal{D}(\boldsymbol{u})$ となる．ところで，$\boldsymbol{s}_n$ はタイプIの接尾辞なので，$\boldsymbol{s}_n \notin \mathcal{D}(\boldsymbol{x}^{n-1})$ である．すなわち，$\boldsymbol{u}$ 上に $\boldsymbol{s}_n$ が出現するから，$\boldsymbol{u} \notin \mathcal{D}(\boldsymbol{x}^{n-1})$ となる．したがって，$\boldsymbol{u} \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)$ である．

次に，$|\boldsymbol{u}| < |\boldsymbol{s}_n|$ の場合を考える．接尾辞 $\boldsymbol{s}_n$ は最も短いタイプIの接尾辞である．よって，$\boldsymbol{u} \notin \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)$ となる．したがって，$\boldsymbol{u} \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ である．

以上より，題意が成立する．

□

系 5 任意の接尾辞 $\boldsymbol{u} \in \mathcal{S}(\boldsymbol{x}^n)$ が，$\boldsymbol{u} \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^n)\,, \sigma(\boldsymbol{u}) \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ ならば，$\boldsymbol{u} = \boldsymbol{s}_n$.

次の命題Aと命題Bで，接尾辞のタイプと $\mathcal{A}_{\mathrm{DEL}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})\,, \mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$ との関係を示す．

命題 A (定理 6.5 [23])

任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，

$$\mathcal{A}_{\mathrm{DEL}}\Big(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\Big) = \{\boldsymbol{s}_{n+1}\}.$$

命題 B (Proposition 3 [24])
任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，系列 $\boldsymbol{w}$ が $\boldsymbol{w} \in \mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$ ならば，

$$\pi(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right) \quad \text{または，} \quad \sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right).$$

なお，両方の命題とも本論文に適した形で表記を変えているが，本質的に文献 [23][24] と同じ結果である．

5.5 新しく加わる極小禁止語の長さ

入力系列が $\boldsymbol{x}^n$ から $\boldsymbol{x}^{n+1}$ に変わったとき，命題 B から，反辞書 $\mathcal{A}(\boldsymbol{x}^n)$ に加える極小禁止語の系列長は $\boldsymbol{x}^{n+1}$ のタイプ I の接尾辞の系列長に関係があることがわかる．そして，タイプ I の接尾辞の系列長は $|\boldsymbol{s}_{n+1}|$ 以上である．

ここで，$\boldsymbol{s}_n$ と $\boldsymbol{s}_{n+1}$ の系列長の関係について，次の補題 8 で示す．

補題 8 任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，

$$|\boldsymbol{s}_{n+1}| \leq |\boldsymbol{s}_n| + 1.$$

証明 命題 5 から，$\sigma(\boldsymbol{s}_{n+1}) \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^{n+1})$ である．よって，$\sigma(\boldsymbol{s}_{n+1}) \in \mathcal{D}(\boldsymbol{x}^n)$．ここで，$\boldsymbol{s}_{n+1}$ の末尾は x_{n+1} だから，$\boldsymbol{u} = \pi(\boldsymbol{s}_{n+1})$ とおくと，$\sigma(\boldsymbol{u})x_{n+1} \in \mathcal{D}(\boldsymbol{x}^n)$ となる．したがって，$\sigma(\boldsymbol{u}) \in \mathcal{D}(\boldsymbol{x}^{n-1})$ である．ここで，$\sigma(\boldsymbol{u})$ は $\boldsymbol{x}^n$ の接尾辞なので，$\sigma(\boldsymbol{u}) \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^n)$ となる．したがって，$|\sigma(\boldsymbol{u})| < |\boldsymbol{s}_n|$ である．すなわち，$|\boldsymbol{u}| = |\sigma(\boldsymbol{u})| + 1 \leq |\boldsymbol{s}_n|$ となる．

以上より，$|\boldsymbol{s}_{n+1}| = |\boldsymbol{u}x_{n+1}| = |\boldsymbol{u}| + 1 \leq |\boldsymbol{s}_n| + 1$ となり，題意が成立する．等号は，$\boldsymbol{u}$ が $\boldsymbol{x}^n$ の接尾辞なので，$\boldsymbol{u} = \boldsymbol{s}_n$ のとき，すなわち，$\boldsymbol{s}_{n+1} = \boldsymbol{s}_n x_{n+1}$ のときに限り成り立つ．

□

さらに，新しく加わる極小禁止語の長さの上限と下限を，定理 5 に示す．

定理 5 任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，系列 $\boldsymbol{w}$ が $\boldsymbol{w} \in \mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$ ならば，

$$|\boldsymbol{s}_{n+1}| + 1 \leq |\boldsymbol{w}| \leq |\boldsymbol{s}_n| + 2.$$

証明 最初に，$|\boldsymbol{s}_{n+1}| + 1 \leq |\boldsymbol{w}|$ を証明する．

命題 B から，$\pi(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ または，$\sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ である．すなわち，$\boldsymbol{s}_{n+1}$ は $\boldsymbol{x}^{n+1}$ のタイプ I の最短の接尾辞だから，$|\pi(\boldsymbol{w})| = |\sigma(\boldsymbol{w})| \geq |\boldsymbol{s}_{n+1}|$ である．したがって，$|\boldsymbol{w}| = |\pi(\boldsymbol{w})| + 1 = |\sigma(\boldsymbol{w})| + 1 \geq |\boldsymbol{s}_{n+1}| + 1$ である．

次に，$|\boldsymbol{w}| \leq |\boldsymbol{s}_n| + 2$ を証明する．

命題 B から，$\pi(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ の場合と，$\pi(\boldsymbol{w}) \notin \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ の場合に分けて証明する．また，表記を簡潔にするために $\boldsymbol{u} = \pi(\sigma(\boldsymbol{w}))$ とおく．ここで，$\boldsymbol{w}$ は $\boldsymbol{x}^{n+1}$ の極小禁止語なので，$\sigma(\boldsymbol{w}) \in \mathcal{D}(\boldsymbol{x}^{n+1})$ である．よって，$\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}^{n})$ である．

まずは，$\pi(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ の場合を考える．このとき，$\boldsymbol{u} = \sigma(\pi(\boldsymbol{w})) \in \mathcal{S}(\boldsymbol{x}^{n+1})$ である．さらに，$\boldsymbol{u} \in \mathcal{D}(\boldsymbol{x}^{n})$ から，$\boldsymbol{u} \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^{n+1})$ である．よって，系 5 から，$\pi(\boldsymbol{w}) = \boldsymbol{s}_{n+1}$ である．このとき，補題 8 から，$|\pi(\boldsymbol{w})| = |\boldsymbol{s}_{n+1}| \leq |\boldsymbol{s}_n| + 1$ である．したがって，$|\boldsymbol{w}| = |\pi(\boldsymbol{w})| + 1 \leq |\boldsymbol{s}_n| + 2$ となる．

今度は，$\pi(\boldsymbol{w}) \notin \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ の場合を考える．このとき，命題 B から，$\sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ である．ここで，もし $\pi(\boldsymbol{w}) \in \mathcal{S}(\boldsymbol{x}^{n})$ と仮定すると，$\sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ から，$\boldsymbol{w} \in \mathcal{S}(\boldsymbol{x}^{n+1})$ となる．すなわち，$\boldsymbol{w} \in \mathcal{D}(\boldsymbol{x}^{n+1})$ である．これは，$\boldsymbol{w} \in \mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^{n} \rightarrow \boldsymbol{x}^{n+1})$ に矛盾する．したがって，$\pi(\boldsymbol{w}) \notin \mathcal{S}(\boldsymbol{x}^{n})$ である．すなわち，$\pi(\boldsymbol{w}) \in \mathcal{D}(\boldsymbol{x}^{n-1})$ となる．

ところで，$\sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ から，$\boldsymbol{u} = \pi(\sigma(\boldsymbol{w})) \in \mathcal{S}(\boldsymbol{x}^{n})$ である．また，$\pi(\boldsymbol{w}) \in \mathcal{D}(\boldsymbol{x}^{n-1})$ から，$\boldsymbol{u} = \sigma(\pi(\boldsymbol{w})) \in \mathcal{D}(\boldsymbol{x}^{n-1})$ となる．よって，$\boldsymbol{u} \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^{n})$ である．したがって，$|\boldsymbol{u}| < |\boldsymbol{s}_n|$ から，$|\boldsymbol{w}| = |\boldsymbol{u}| + 2 < |\boldsymbol{s}_n| + 2$ である．

以上より，題意が成立する．等号は，$\boldsymbol{s}_{n+1} = \boldsymbol{s}_n x_{n+1}$ のときに限り成り立つ．

□

なお，補題 8 と定理 5 の等号条件 $\boldsymbol{s}_{n+1} = \boldsymbol{s}_n x_{n+1}$ には $\boldsymbol{s}_{n+1}$ と x_{n+1} が使われている．そのため，素朴な方法では x_{n+1} の入力後でないと等号条件を満たすかどうかわからない．

だが，反辞書を用いることで，x_{n+1} の入力前に，どの記号が x_{n+1} になれば等号条件を満たすのかがわかることを，次の命題 6 で示す．

命題 6 任意の系列 $\boldsymbol{x}^{n}$ $(n \geq 1)$ に対して，$\boldsymbol{s}_n x_{n+1} \in \mathcal{A}(\boldsymbol{x}^{n})$ ならば，

$$\boldsymbol{s}_{n+1} = \boldsymbol{s}_n x_{n+1}.$$

証明 系列 $\boldsymbol{s}_n x_{n+1}$ は $\boldsymbol{x}^{n}$ の極小禁止語だから，$\boldsymbol{s}_n x_{n+1} \notin \mathcal{D}(\boldsymbol{x}^{n})$ である．したがって，$\boldsymbol{s}_n x_{n+1} \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ である．また，同じく $\boldsymbol{s}_n x_{n+1} \in \mathcal{A}(\boldsymbol{x}^{n})$ より，$\sigma(\boldsymbol{s}_n) x_{n+1} \in \mathcal{D}(\boldsymbol{x}^{n})$ である．すなわち，$\sigma(\boldsymbol{s}_n) x_{n+1} \in \mathcal{S}_{\mathrm{II}}(\boldsymbol{x}^{n+1})$ となる．

よって，系 5 より，題意は成立する．

□

5.6 反辞書の変化と提案アルゴリズム

反辞書を動的に構築する手法は，これまでの私たちの研究成果 [23], [24] で提案している．しかし，従来の反辞書構築手法では，静的な手法でも動的な手法でも，入力記号系列から接尾辞トライや接尾辞木や接尾辞配列といった接尾辞のデータ構造を構築し，それから反辞書を構築している．

本論文では，この接尾辞のデータ構造の構築を省き，入力系列から直接反辞書オートマトンを構築する手法を提案する．

まず, 命題Bにより, 新しく加わる極小禁止語は2つのタイプ $\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$, $\mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ に分けることができる．

$$\begin{aligned}\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right) &= \left\{\boldsymbol{w}\in\mathcal{A}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right)|\pi(\boldsymbol{w})\in\mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right)\right\},\\ \mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right) &= \left\{\boldsymbol{w}\in\mathcal{A}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right)|\sigma(\boldsymbol{w})\in\mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right)\right\},\end{aligned}$$

なお, 両方のタイプに属する極小禁止語が存在する事もあるが, $\mathcal{A}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}) = \mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})\cup\mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ であることを注記する．

また，$\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ に属する極小禁止語を $\boldsymbol{x}^{n+1}$ のタイプIの極小禁止語と，$\mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ に属する極小禁止語を $\boldsymbol{x}^{n+1}$ のタイプIIの極小禁止語と呼ぶことにする．

さらに，定理 1 と系 1 の反辞書の別表現と組み合わせると，

$$\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right) = \left\{\boldsymbol{u}\beta|\beta\in\mathcal{T}_{n+1}(\boldsymbol{u}),\boldsymbol{u}\in\mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right)\right\}, \tag{5.6}$$

$$\mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1}\right) = \left\{\alpha\boldsymbol{u}|\alpha\in\mathcal{H}_{n+1}(\boldsymbol{u}),\boldsymbol{u}\in\mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right)\right\} \tag{5.7}$$

となる．

ここで，タイプIの極小禁止語について，以下の補題 9 と定理 6 を示す.

補題 9 任意の系列 $\boldsymbol{w}$ に対して，$\boldsymbol{w}\in\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ ならば,

$$\pi(\boldsymbol{w}) = \boldsymbol{s}_{n+1}.$$

証明 表記を簡潔にするために，$\boldsymbol{u}=\pi(\boldsymbol{w})$ とおき，仮定法を用いて証明する．

もし，$\boldsymbol{u}\neq\boldsymbol{s}_{n+1}$ と仮定したとき，$\boldsymbol{u}\in\mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ と命題 5 から，$\sigma(\boldsymbol{u})\in\mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})$ である．よって，$\mathcal{X}^{\mathrm{T}}_{n+1}(\boldsymbol{u}) = \mathcal{X}^{\mathrm{T}}_{n+1}(\sigma(\boldsymbol{u})) = \phi$ となる．また，$\boldsymbol{u}\neq\lambda$. したがって，$\mathcal{T}_{n+1}(\boldsymbol{u}) = \mathcal{X}^{\mathrm{T}}_{n+1}(\sigma(\boldsymbol{u}))\setminus\mathcal{X}^{\mathrm{T}}_{n+1}(\boldsymbol{u}) = \phi$ になる．これは，$\boldsymbol{w}\in\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}(\boldsymbol{x}^n\rightarrow\boldsymbol{x}^{n+1})$ に矛盾する．よって，$\boldsymbol{u}=\boldsymbol{s}_{n+1}$ である．

□

定理 6 任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，

$$\mathcal{A}^{\mathrm{I}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right) \;=\; \{\boldsymbol{s}_{n+1}\beta | \beta \in \mathcal{T}_{n+1}(\boldsymbol{s}_{n+1})\}\,.$$

証明 補題 9 と (5.6) から，明らかである．

□

次に，タイプ II の極小禁止語について，以下の定理 7 を示す．

定理 7 任意の系列 $\boldsymbol{x}^n$ $(n \geq 1)$ に対して，

$$\mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}\left(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1}\right) = \left\{\alpha\boldsymbol{u} | \alpha \in \mathcal{H}_{n+1}(\boldsymbol{u})\,, \boldsymbol{u} \in \mathcal{S}_{\mathrm{I}}\left(\boldsymbol{x}^{n+1}\right), |\boldsymbol{s}_{n+1}| \leq |\boldsymbol{u}| \leq |\boldsymbol{s}_n| + 1\right\}.$$

証明 定理 5 と (5.7) から，明らかである．

□

ところで，系列 $\boldsymbol{x}^n$ の最短のタイプ I の接尾辞 $\boldsymbol{s}_n$ は，$\boldsymbol{x}^n$ の反辞書オートマトンのノード $\mathcal{N}(\boldsymbol{x}^n)$ のラベル列と一致する．また，命題 A から，$\boldsymbol{s}_{n+1}$ は MFW ノード $\mathcal{N}(\boldsymbol{x}^n x_{n+1})$ のラベル列である．この MFW ノードを p とおく．定理 6 を用いると，p の suffix link 先ノード $\mathcal{S}(p)$ から記号 β で遷移した先が中間ノードのとき，$\boldsymbol{L}(p)\beta$ は $\boldsymbol{x}^{n+1}$ のタイプ I の極小禁止語である．

一方，$\boldsymbol{x}^{n+1}$ のタイプ II の極小禁止語 $\boldsymbol{w} \in \mathcal{A}^{\mathrm{II}}_{\mathrm{NEW}}(\boldsymbol{x}^n \rightarrow \boldsymbol{x}^{n+1})$ は，定理 7 から，$\sigma(\boldsymbol{w}) \in \mathcal{S}_{\mathrm{I}}(\boldsymbol{x}^{n+1})\,, |\boldsymbol{s}_{n+1}| \leq |\sigma(\boldsymbol{w})| \leq |\boldsymbol{s}_n| + 1$ である．さらに，命題 5 から，$\sigma(\boldsymbol{w})$ の末尾に $\boldsymbol{s}_{n+1}$ が出現する．すなわち，$\boldsymbol{w}$ を $\boldsymbol{x}^n$ の反辞書オートマトンに入力すると，p に到達する．したがって，ルートから p へ至る全てのパスから $\boldsymbol{w}$ を見つければよい．

以上のことから，新しい記号 x_{n+1} が入力されたときに反辞書オートマトンを変化させるアルゴリズム change_AAM を以下に提案する．

このアルゴリズムは，既に $\boldsymbol{x}^n$ の反辞書オートマトンが構築済みであるものとする．また，$\boldsymbol{x}^0 = \lambda$ の反辞書オートマトンは，初期化作業により与えられるものとする．

Algorithm change_AAM
begin 1
  $p \leftarrow \mathcal{N}(\boldsymbol{s}_n x_{n+1})$ 2
  p からタイプ I の MFW ノードを伸ばす. 3
  for ($\forall \boldsymbol{w}$: p に至るパスから生成した系列) **do** 4
    if ($|\boldsymbol{s}_{n+1}| \leq |\sigma(\boldsymbol{w})| \leq |\boldsymbol{s}_n| + 1$) 5

$\boldsymbol{w}$ をラベル列に持つ新しいMFWノードを追加する.

end.

第6章　おわりに

反辞書は，接尾辞木や接尾辞配列と同様に，入力系列の特徴が反映されている．そして，反辞書オートマトンにより，計算機への負荷が少なく高速に動作するという利点がある．データ圧縮などでも，優れた圧縮性能を持っている．しかし，反辞書構築による計算機への負荷がとても大きいため，扱いずらい面もある．

そこで，本論文では，効率的に反辞書を構築する手法を検討した．

得られた結果は，以下のとおりである．

第4章では，接尾辞配列を用いた静的な反辞書構築手法を提案した．検討の過程では，反辞書の別表現を新たに提案し，極小禁止語を，先頭記号と後続する系列に分けて考えることにより，あと1記号先頭に加えることで極小禁止語になる系列が，入力系列の接尾辞配列ならびにL配列によって特徴づけられること，すなわち，先頭に１記号加えれば極小禁止語になる系列は，接尾辞配列上の接尾辞の先頭部分に存在し，その長さはL配列で求めることができることを明らかにした．これにより，線形時間での構築が可能となった．実験結果では，接尾辞木を用いた構築手法と比べて，記憶量と計算量を改善することができた．

第5章では，動的に反辞書オートマトンを構築する手法を提案した．従来の反辞書構築手法では，静的手法でも動的手法でも，入力記号系列から接尾辞トライや接尾辞木や接尾辞配列などを構築し，それから反辞書を構築する．そこで，本論文では，入力系列の末尾に新たな記号が加わるたびに，逐次的に反辞書オートマトンを変化させ，入力系列から直接反辞書を構築する手法を提案する．

そのために，入力系列の末尾に新しい記号が追加されることで，反辞書にどのような変化が起きるのかを明らかにした．これにより，接尾辞のデータ構造を経由しないで反辞書オートマトンを構築することが可能となった．

今後の研究課題としては，この成果を活用した応用技術を開発することである．

謝辞

本研究を御指導して下さった電気通信大学情報システム学研究科 森田 啓義 教授には，研究が遅々として進まないだけでなく，執筆も遅い私へ親身に助言と御指導をして戴きました．そして，研究方法と研究者の心構えについても多くのことを学びました．このことに深く感謝しお礼申し上げます.

また，本研究を通してとても貴重な助言を戴きました長野県工科短期大学校 太田 隆博 准教授にも，多くの迷惑を掛けながらも，親身に助言を戴きました．心より厚くお礼申し上げます.

そして，研究のきっかけを与えてくださった職業能力開発総合大学校の松嶋 智子 准教授，研究しやすい環境づくりに影から支えてくださった高知職業能力開発短期大学校の松中 孝二 校長をはじめとする職場の同僚に感謝いたします.

最後に，何年間も高知と東京の往復をすることを支えてくれ，ときには挫折しそうなときに叱咤激励してくれた家族に深く深く感謝いたします.

ありがとうございました．

関連論文リスト

■学術雑誌論文（査読あり）

[1] 深江 裕忠, 太田 隆博, 森田 啓義 : ”配列を用いた反辞書の線形構築法”, 電子情報通信学会和文論文誌 A Vol.J97-A,No.1, pp.33–44 (2014).【第 4 章と関係】

■国際会議発表（査読あり）

[2] H. Fukae, T. Ota and H. Morita : ”On fast and memory-efficient construction of an antidictionary array”, Proc. of 2012 IEEE Int’l Symp. on Information Theory (ISIT2012), Cambridge, pp.1092–1096 (2012). 【第 4 章と関係】

■口頭発表

[3] 深江 裕忠, 太田 隆博, 森田 啓義 : ”接尾辞配列による反辞書構築法”, 第 33 回情報理論とその応用シンポジウム (SITA2010), 松代, pp.829–834 (2010).【第 4 章と関係】

[4] 深江 裕忠, 森田 啓義 : ”Suffix array を利用した動的な反辞書生成”, 第 30 回情報理論とその応用シンポジウム (SITA2007), 賢島, pp.384–387 (2007).【第 5 章と関係】

参考文献

[5] M. Crochemore, F. Mignosi, A. Restivo and S. Salemi : ”Data compression using antidictionaries”, Proc. IEEE, 88, 11, pp.1756–1768 (2000).

[6] 西新, 森田, 太田 : ”反辞書木を用いた分岐予測手法”, 電子情報通信学会論文誌 A Vol.J93-A No.3, pp.204–206 (2010).

[7] D. Sun, H. Morita and M. Nishiara : ”On construction of reversible variable-length codes including synchronization markers as codewords”, Proc. of 2006 Int'l Symp. on Information Theory and its Applications (ISITA2006), pp.12–16 (2006).

[8] 太田, 戸松, 木村, 森田 : ”反辞書符号化法を用いた不整脈検出”, 信学技報 MBE2010-7, pp.35–40 (2010).

[9] T. Ota and H. Morita : ”On the construction of an antidictionary with linear complexity using the suffix tree”, IEICE Trans. on Fundamentals, E90-A, 11, pp.2533–2539 (2007).

[10] U. Manber and G. Myers : ”Suffix arrays: A new method for on-line string searches”, SIAM Journal of Computing, 22(5), pp.935–948 (1993).

[11] 笠井 : ”部分語計数問題の接尾辞配列を用いた高速アルゴリズム”, Master's thesis, 九州大学大学院システム情報科学研究科 (1999).

[12] T. Kasai, G. Lee, H. Arimura, S. Arikawa and K. Park : ”Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications”, Proc. of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM 2001), Lecture Notes in Computer Science, 2089, pp.181–192 (2001).

[13] A. Aho ほか著, 大野義夫 訳 : ”データ構造とアルゴリズム”, 培風館 (1987).

[14] D. Gusfield : ”Algorithms on strings, trees, and sequences”, Cambridge Univ.Press (1997).

[15] E. Ukkonen : ”On-line construction of suffix-trees”, Algorithmica, 14, pp.249–260 (1995).

[16] G. Nong, S. Zhang and W. H. Chan : "Linear Suffix Array Construction by Almost Pure Induced-Sorting", Proc. of 19th IEEE Data Compression Conference (2009).

[17] J. Kärkkäinen and P. Sanders : "Simple Linear Work Suffix Array Construction", Proc. of 30th International Conference on Automata, Languages and Programming, pp.943-955 (2003).

[18] J. Kärkkäinen, P. Sanders and S. Burkhardt : "Linear Work Suffix Array Construction", J.ACM 53 (6), pp.918–936 (2006).

[19] D. K. Kim, J. S. Sim, H. Park and K. Park : "Constructing Suffix Arrays in Linear Time", J.Discrete Algorithms 3(2-4), pp.126–142 (2005).

[20] P. Ko and S. Aluru : "Space Efficient Linear Time Construction of Suffix Arrays", J.Discrete Algorithms 3(2-4), pp.143–156 (2005).

[21] R. Sedgewick : "Algorithms in C", Addison-Wesley (1990).

[22] T. Asano, S. Bereg and D. Kirkpatrick : "Finding Nearest Larger Neighbors – A Case Study in Algorithm Design and Analysis –", Lecture Notes in Computer Science, 5760, pp.249–260 (2009).

[23] 深江 : "接尾辞配列を利用した動的反辞書生成に関する研究", Master's thesis, 電気通信大学大学院情報システム学研究科 (2008).

[24] T. Ota, H. Morita and H. Fukae : "On the Dynamic Construction of an Antidictionary with Linear Complexity", Proc. of 2011 IEEE Int'l Symp. on Information Theory (ISIT2011), pp.2422–2426 (2011).